精通
HTML 5
网页设计

一册在手·全面复制

- 全面介绍HTML 5网页制作技术
- 基础知识结合示例讲解，综合案例培养实践能力
- 时长达600分钟教学视频、代码、课件提供云下载



刘玉萍 刘增杰 编著

清华大学出版社

精通 HTML 5 网页设计

刘玉萍　刘增杰 编著

清华大学出版社
北　京

内 容 简 介

本书循序渐进地介绍了 HTML 5 网页设计的基础知识和技能，提供了大量的 HTML 5 应用实例供读者实践，并清晰阐述代码如何工作方式及作用，使读者能在最短的时间内掌握 HTML 5 的应用技能。

全书共分 20 章，分别介绍 HTML 5 基本概念、HTML 5 中的新增元素、认识网页和网站、HTML 5 中的文档结构、HTML 5 中的文本和图像、使用 HTML 5 建立超链接、使用 HTML 5 建立表格、使用 HTML 5 建立表单、使用 HTML 5 绘制图像、HTML 5 中的音频和视频、如何在 HTML 5 获取地理位置、Web 通信新技术、本地存储技术、线程处理技术、构造离线 Web 应用程序、使用 CSS3 修饰网页样式、JavaScript 快速入门和 HTML 5、CSS3 和 JavaScript 的搭配应用等，最后通过开发两个综合案例——企业门户网站和 HTML 5 游戏，使读者进一步巩固所学的知识，提高综合实战能力。

本书适合 HTML 5 网页设计初学者、网站设计人员和网站开发人员使用。

图书在版编目（CIP）数据

精通 HTML5 网页设计/刘玉萍，刘增杰编著. —北京：清华大学出版社，2013.
ISBN 978-7-302-33621-1

Ⅰ.①精… Ⅱ.①刘… ②刘… Ⅲ.①超文本标记语言—程序设计 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字（2013）第 203878 号

责任编辑：夏非彼
封面设计：王　翔
责任校对：闫秀华
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　张：**21.75　　**字　数：**557 千字
版　　次：2013 年 10 月第 1 版　　**印　次：**2013 年 10 月第 1 次印刷
印　　数：1~3000 册
定　　价：49.00 元

产品编号：047621-01

前　言

HTML 5 是用于取代 HTML 4.01 和 XHTML 1.0 标准的 HTML 标准版本，现在仍处于发展阶段，但大部分浏览器已经开始支持 HTML 5 技术。HTML 5 有两大特点：一是强化了 Web 网页的表现能力，另一是增加了离线存储、Web 通讯等等新功能。随着 HTML 5 规范的日益成熟，HTML 5 将成为主流页面语言。在这个背景下，目前学习和关注 HTML 5 网页设计的人越来越多，而很多 HTML 5 的初学者都苦于找不到一本通俗易懂、容易入门和案例实用的参考书。为此，作者组织有丰富经验的开发人员写作了这本精通 HTML 5 网页设计的教材。

本书特色

内容全面：知识点由浅入深，涵盖了所有 HTML 5 网页设计知识点，读者可以由浅入深地掌握 HTML 5 网页设计技术。

图文并茂：注重操作，图文并茂，在介绍案例的过程中，每一个操作均有对应的插图。这种图文结合的方式使读者在学习过程中能够直观、清晰地看到操作的过程以及效果，便于更快地理解和掌握。

易学易用：颠覆传统“看”书的观念，变成一本能“操作”的图书。

案例丰富：把知识点融汇于系统的案例实训当中，并且结合经典案例进行讲解和拓展。进而达到“知其然，并知其所以然”的效果。

提示技巧：本书对读者在学习过程中可能会遇到的疑难问题以“提示”和“技巧”的形式进行了说明，以免读者在学习的过程中走弯路。

技术实用：本书所有案例都是模仿现实网站开发而设计，在实际工作中的参考性比较强。

读者对象

本书是一本完整介绍 HTML 5 网页设计的教程，适合如下读者学习使用：

- 对网页设计感兴趣的读者，可以快速上手设计精美的网页。
- 对 HTML 5 语言感兴趣的读者，想快速掌握 HTML 5 并设计需要的网页。
- 开发 Web 系统的人，是独一无二的参考。

代码、课件、教学视频下载与技术支持

本书代码、课件可以从下面网址下载：

http://pan.baidu.com/share/link?shareid=1809634813&uk=3056331768

教学视频可以从下面网址下载：

http://pan.baidu.com/share/link?shareid=1852687522&uk=3056331768

如果下载有问题，请发邮件给 booksaga@163.com，邮件主题为“求代码，精通 HTML5”。

致谢

参与本书编写人员除了封面署名人员以外，还有张俊斌、王英英、吴志才、姬远鹏、张少军、苏士辉、肖品、孙若淞、宋冰冰、王攀登、张工厂、王婷婷、王维维、梁云亮、陈伟光、臧顺娟、程铖、卢健良、李亚飞、李清海、王鹏程、王雪涛、邴万强、陈彭涛、冯玲、郭丽娟、钱东省、王飞、原杨、李坤、武炜、史艳艳、包惠利等人。由于作者水平有限，时间仓促，书中难免有错漏之处，敬请读者谅解。

编　者
2013 年 8 月

目　　录

第 1 章　HTML 5 概述

目前，网络已经成为人们生活工作当中不可缺少的一部分，那么网页就是呈现给人们信息的平台。因此，怎么样把信息很好地呈现在网页当中，就成了一个值得研究的课题——网页设计与制作。制作网页可采用可视化编辑软件，但是无论采用哪一种网页编辑软件，最后都要将所设计的网页转化为 HTML 语言，当前最新版本是 HTML 5。

1.1　HTML 5 简介

HTML 是用来描述网页的一种语言——标记语言（标记语言是一套标记标签，HTML 使用标记标签来描述网页），而不是编程语言，它是制作网页的基础语言，主要用来描述超文本中内容的显示方式。

1.1.1　HTML

标记语言经过浏览器的解释和编译，能正确显示 HTML 标记的内容，但它本身不显示在浏览器中。

HTML 语言从 HTML 1.0 至 HTML 5 经历了巨大地变化，从单一的文本显示功能到图文并茂的多媒体显示功能，许多特性经过多年地完善，已经成为一种非常标准的标记语言。HTML 经历的版本及发布日期如下表所示。

版本	发布日期	说明
超文本标记语言（第一版）	1993 年 6 月	作为互联网工程工作小组（IETF）工作草案发布（并非标准）
HTML 2.0	1995 年 11 月	作为 RFC 1866 发布，在 RFC 2854 发布之后被宣布已经过时
HTML 3.2	1996 年 1 月 14 日	W3C 推荐标准
HTML 4.0	1997 年 12 月 18 日	W3C 推荐标准
HTML 4.01	1999 年 12 月 24 日	微小改进，W3C 推荐标准
ISO HTML	2000 年 5 月 15 日	基于严格的 HTML 4.01 语法，是国际标准化组织和国际电工委员会的标准
XHTML 1.0	2000 年 1 月 26 日	W3C 推荐标准，后来经过修订于 2002 年 8 月 1 日重新发布
XHTML 1.1	2001 年 5 月 31 日	较 1.0 有微小改进

（续表）

版本	发布日期	说明
XHTML 2.0 草案	没有发布	2009 年，W3C 停止了 XHTML 2.0 工作组的工作
HTML 5 草案	2008 年 1 月	目前的 HTML 5 规范都是以草案发布，都不是最终版本

完整的 HTML 文档可以说就是一个网页，在该文档中包括 HTML 标签和纯文本，而 Web 浏览器的作用就是读取 HTML 文档，并以网页的形式显示出来。

【例 1.1】一个完整的 HTML 文档

```
<html>
<body>
<h1>标题</h1>
<p>锻炼.</p>
</body>
</html>
```

该 HTML 文档的介绍如下：

- <html>与</html>之间的文本是描述网页的。
- <body>与</body>之间的文本是可见的页面内容。
- <h1>与</h1>之间的文本被显示为标题。
- <p>与</p>之间的文本被显示为段落。

1.1.2 HTML 5

HTML 5 用于取代 1999 年所制定的 HTML 4.01 和 XHTML 1.0，现仍处于发展阶段，但大部分浏览器已经支持 HTML 5 某些技术了。HTML 5 对多媒体的支持功能很强大，新增了如下功能：

- 语义化标签，使文档结构明确。
- 文档对象模型（DOM）。
- 实现 2D 绘图的 Canvas 对象。
- 可控媒体播放。
- 离线存储。
- 文档编辑。
- 拖放。
- 跨文档消息。
- 浏览器历史管理。
- MIME 类型和协议注册。



对于这些新功能，支持 HTML 5 的浏览器在处理 HTML 5 代码出错时必须要更加灵活，而那些不支持 HTML 5 的浏览器将忽略 HTML 5 代码。

HTML 5 的最大优势是语法结构非常简单，它具有以下特点：

- HTML 5 编写简单。即使用户没有任何编程经验，也可以轻松使用 HTML 设计网页，只需将文本加上一些标记（Tags）即可。
- HTML 标记数目有限。在 W3C 所建议使用的 HTML 5 规范中，所有的控制标记都是固定的，且数目也是有限的。所谓固定是指控制标记的名称固定不变，且每个控制标记都已被定义过，其所提供的功能与相关属性的设置都是固定的。这是因为 HTML 中只能引用 Strict DTD、Transitional DTD 或 Frameset DTD 中的控制标记，且 HTML 并不允许网页设计者自行创建控制标记，所以控制标记的数目是有限的，设计者在充分了解每个控制标记的功能后，就可以设计 Web 页面了。
- HTML 语法较弱。在 W3C 制定的 HTML 5 规范中，对于 HTML 5 在语法结构上的规格限制是较松散的，如<HTML>、<Html>或<html>在浏览器中具有同样的功能，是不区分大小写的。另外，也没有严格要求每个控制标记都要有相对应的结束控制标记，如标记<tr>时，不一定需要结束标记</tr>。

HTML 5 最基本的语法是<标记符></标记符>。标记符通常都是成对使用的，有一个开头标记和一个结束标记。结束标记只是在开头标记的前面加一个斜杠“/”。当浏览器收到 HTML 文件后，就会解释里面的标记符，然后把标记符相对应的功能表达出来。

1.1.3 HTML 5 文件的基本结构

完整的 HTML 文件包括标题、段落、列表、表格、绘制的图形以及各种嵌入对象，这些对象统称为 HTML 元素。

HTML 5 文件的基本结构如下：

```
<!DOCTYPE html>
<html >文件开始的标记
<head>文档头部开始的标记
…文件头的内容
</head>文档头部开始的标记
<body>文件主体开始的标记
…文档主体内容
</body>文件主体结束的标记
</html>文件结束的标记
```

从上面的代码可以看出，在 HTML 文件中，所有标记都是对应的，开头标记为“<>”，结束标记为“</>”，在这两个标记中间添加内容，这些基本标记的使用方法及详细解释会在下面的章节呈现。

1.2 HTML 5 文件的编写方法

HTML 5 文本的编写方法有两种，分别如下：

- 手工编写 HTML 文件。
- 使用 HTML 编辑器。

1.2.1　手工编写 HTML 5

由于 HTML 5 是一种标记语言，主要以文本形式存在，因此，所有的记事本工具都可以作为它的开发环境。HTML 文件的扩展名为.html 或.htm，将 HTML 源代码输入到记事本并保存之后，可以在浏览器中打开文档以查看基效果。

【例 1.2】使用记事本编写 HTML 文件

使用记事本编写 HTML 文件的具体操作步骤如下：

01 单击 Windows 桌面上的“开始”按钮，选择“所有程序”→“附件”→“记事本”命令，打开一个记事本，在记事本中输入 HTML 5 代码，如图 1-1 所示。

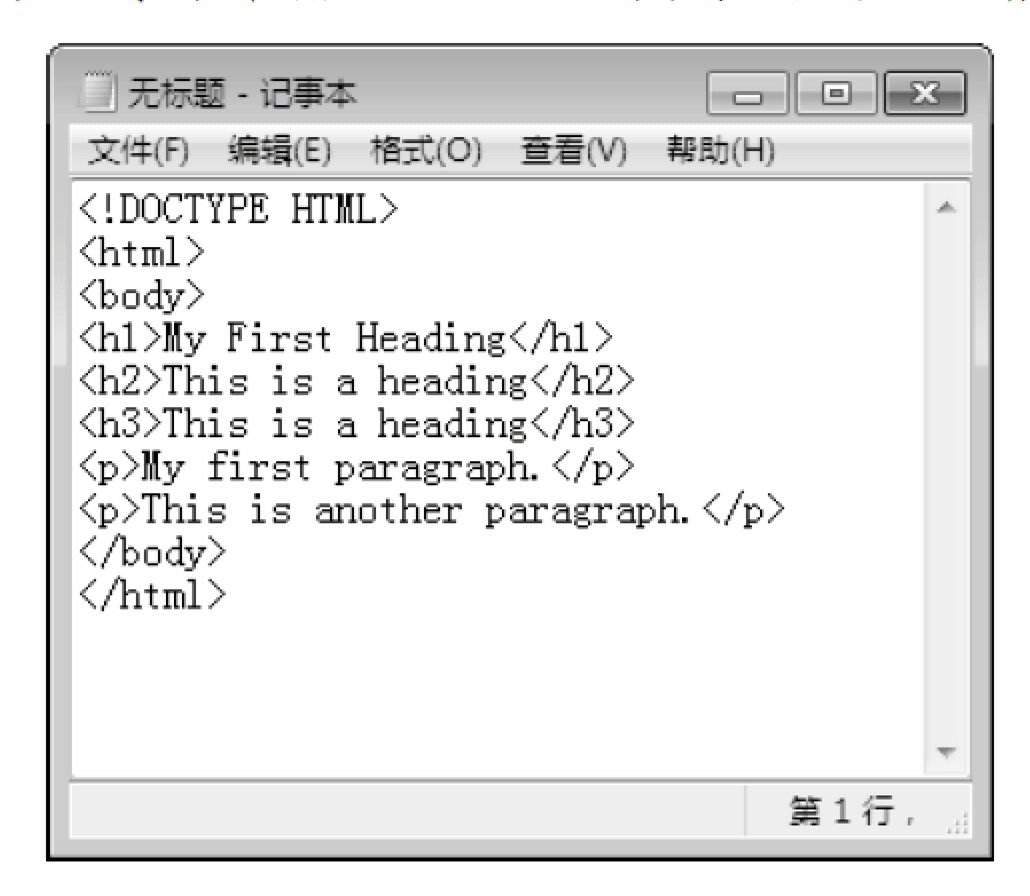


图 1-1　编辑 HTML 代码

02 编辑完 HTML 5 文件后，选择“文件”→“保存”命令→或按 Ctrl+S 快捷键，在弹出的“另存为”对话框中，选择“保存类型”为“所有文件”，然后将文件扩展名设为.html 或.htm，如图 1-2 所示。

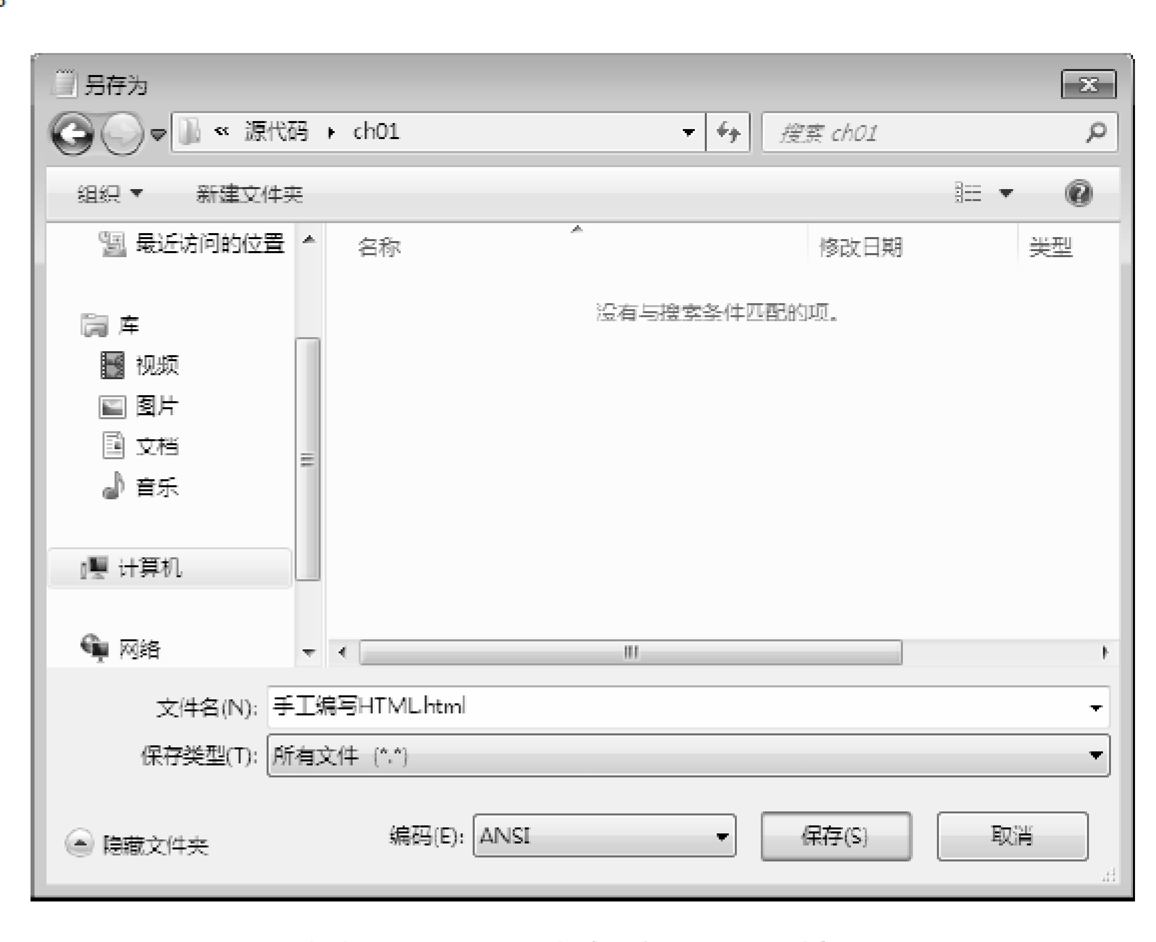


图 1-2　“另存为”对话框

03 单击“保存”按钮保存文件。打开网页文档，在浏览器中预览效果，如图 1-3 所示。

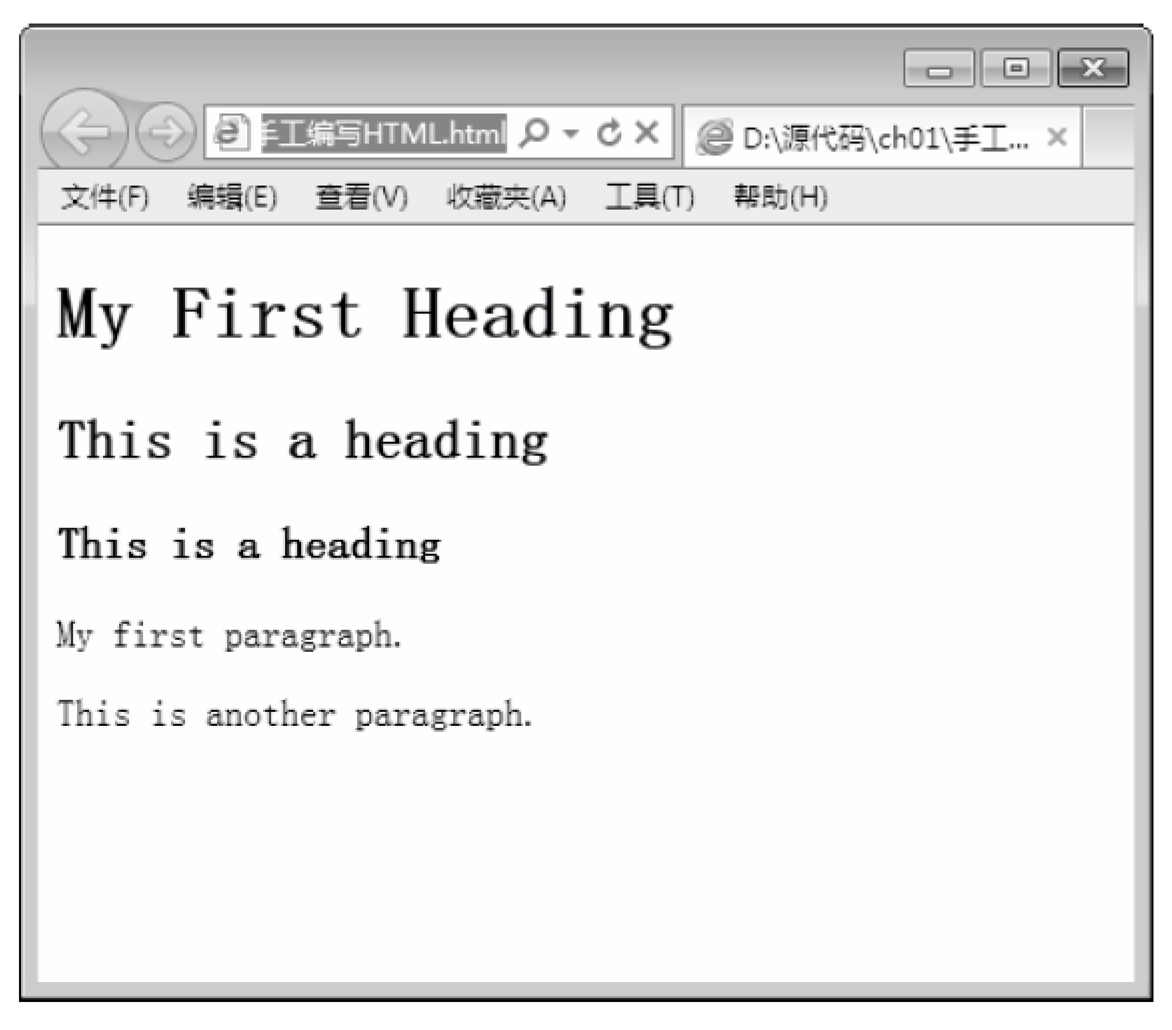


图 1-3　网页的浏览效果



使用记事本可以编写 HTML 文件，但是编写效率太低，对于语法错误及格式都没有提示。

1.2.2 使用 HTML 编辑器

使用 HTML 编辑器可以弥补记事本编写 HTML 文件的缺陷，目前，有很多专门编辑 HTML 网页的编辑器，其中，Adobe 公司出品的 Dreamweaver CS5.5 用户界面非常友好，是一款非常优秀的网页开发工具，并深受广大用户的喜爱。Dreamweaver CS5.5 的主界面如图 1-4 所示。

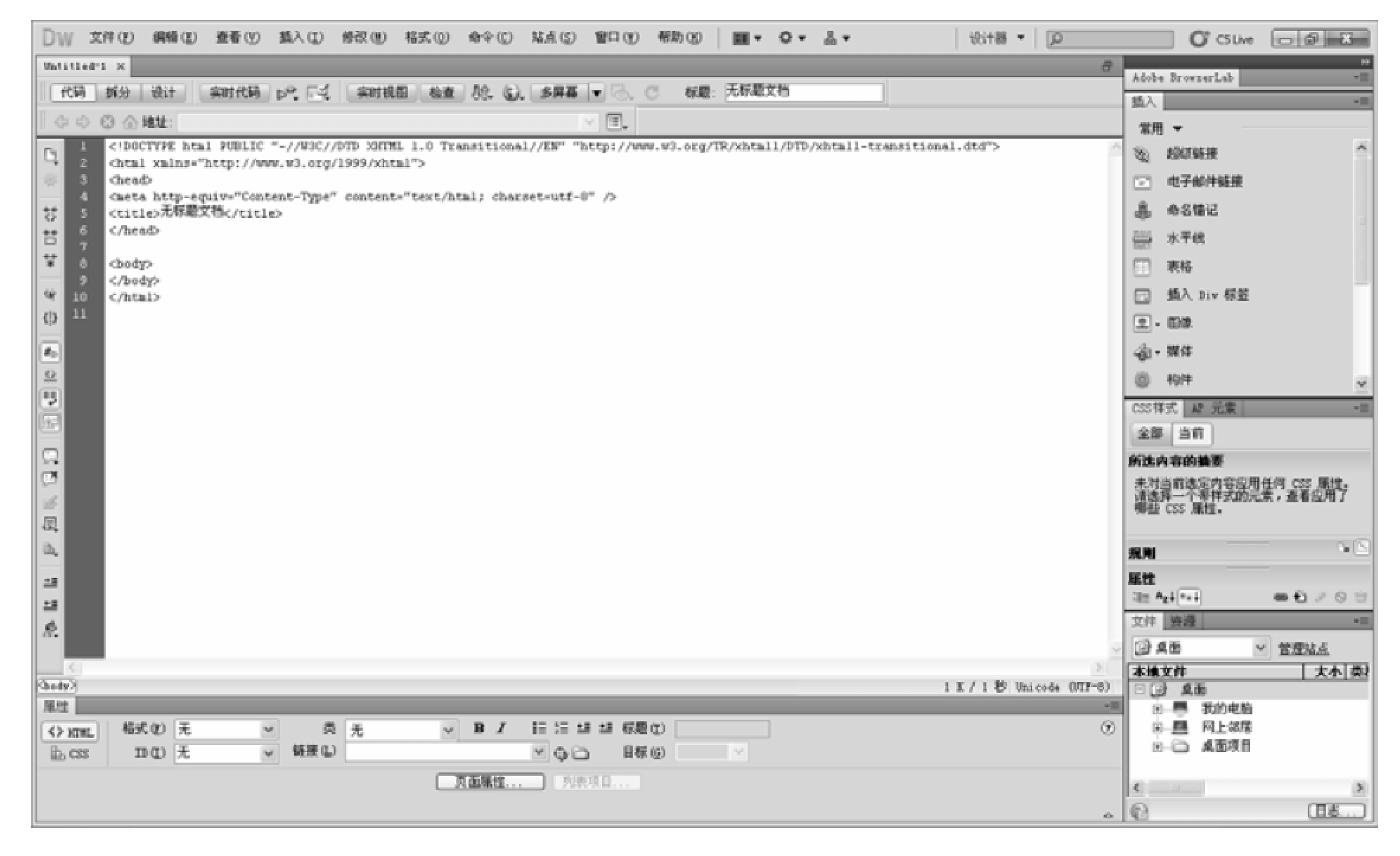


图 1-4　Dreamweaver CS5.5 的主界面

1．文档窗口

文档窗口位于界面的中部，它是用来编排网页的区域，与在浏览器中的结果相似。在文档窗口中，可以将文档分为三种视图显示模式。

- 代码视图。使用代码视图，可以在“文档”窗口中显示当前文档的源代码，也可以在该窗口中直接键入 HTML 代码。
- 设计视图。在设计视图中，无需编辑任何代码，直接使用可视化的操作编辑网页。
- 拆分视图。在拆分视图中，左半部分显示代码视图，右半部分显示设计视图。在这种视图模式下，可以通过键入 HTML 代码，直接观看效果，还可以通过设计视图插入对象，直接查看源文件。

在各种视图间切换，只需在文档工具栏中单击相应的视图按钮即可，文档工具栏，如图 1-5 所示。

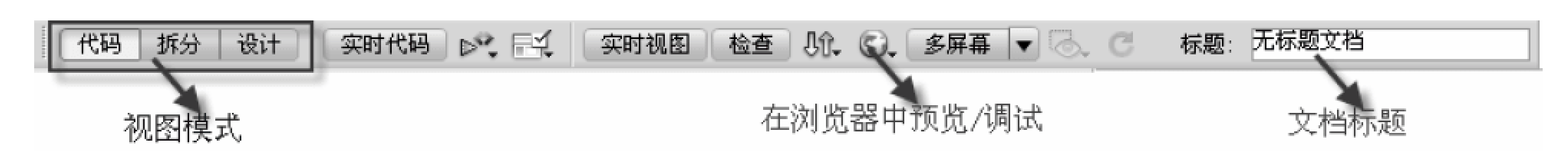


图 1-5　文档工具栏

2．插入面板

插入面板是在设计视图下，使用频度很高的面板之一。插入面板默认打开的是“常用”选项，它包括了最常用的一些对象，例如，在文档中的光标位置插入一段文本、图像或表格等。用户可以根据需要切换到其他选项，如图 1-6 所示。



图 1-6　插入面板包含的页

3．“属性”面板

“属性”面板中主要包含当前选择的对象相关属性设置。可以通过单击菜单栏中的“窗口”→“属性”命令或组合键 Ctrl+F3，打开或关闭“属性”面板。

“属性”面板是常用的面板，因为无论要编辑哪个对象的属性，都要用到它。其内容也会随着选择对象的不同而改变，例如，当光标定位在文档体文字内容部分时，“属性”工具栏显示文字相关属性，如图 1-7 所示。



图 1-7　文字对象的“属性”面板

Dreamweaver CS5.5 中还有很多面板，在以后使用时，再作详细讲解。打开的面板越多，编辑文档的区域会越小，为了方便编辑文档，可以通过 F4 功能键快速隐藏和显示所有面板。

【例 1.3】使用 Dreamweaver CS5.5 编写 HTML 文件

具体操作步骤如下：

01 启动 Dreamweaver CS5.5，如图 1-8 所示，在“欢迎”对话框的“新建”选项组中选择“HTML”选项。或者单击菜单中的“文件”→“新建”命令（快捷键 Ctrl+N）。



图 1-8　包含欢迎屏幕的主界面

02 弹出“新建文档”对话框，如图 1-9 所示，在页面类型选项组中选择 HTML。



图 1-9　“新建文档”对话框

03 单击创建“创建”按钮，创建 HTML 文件，如图 1-10 所示。

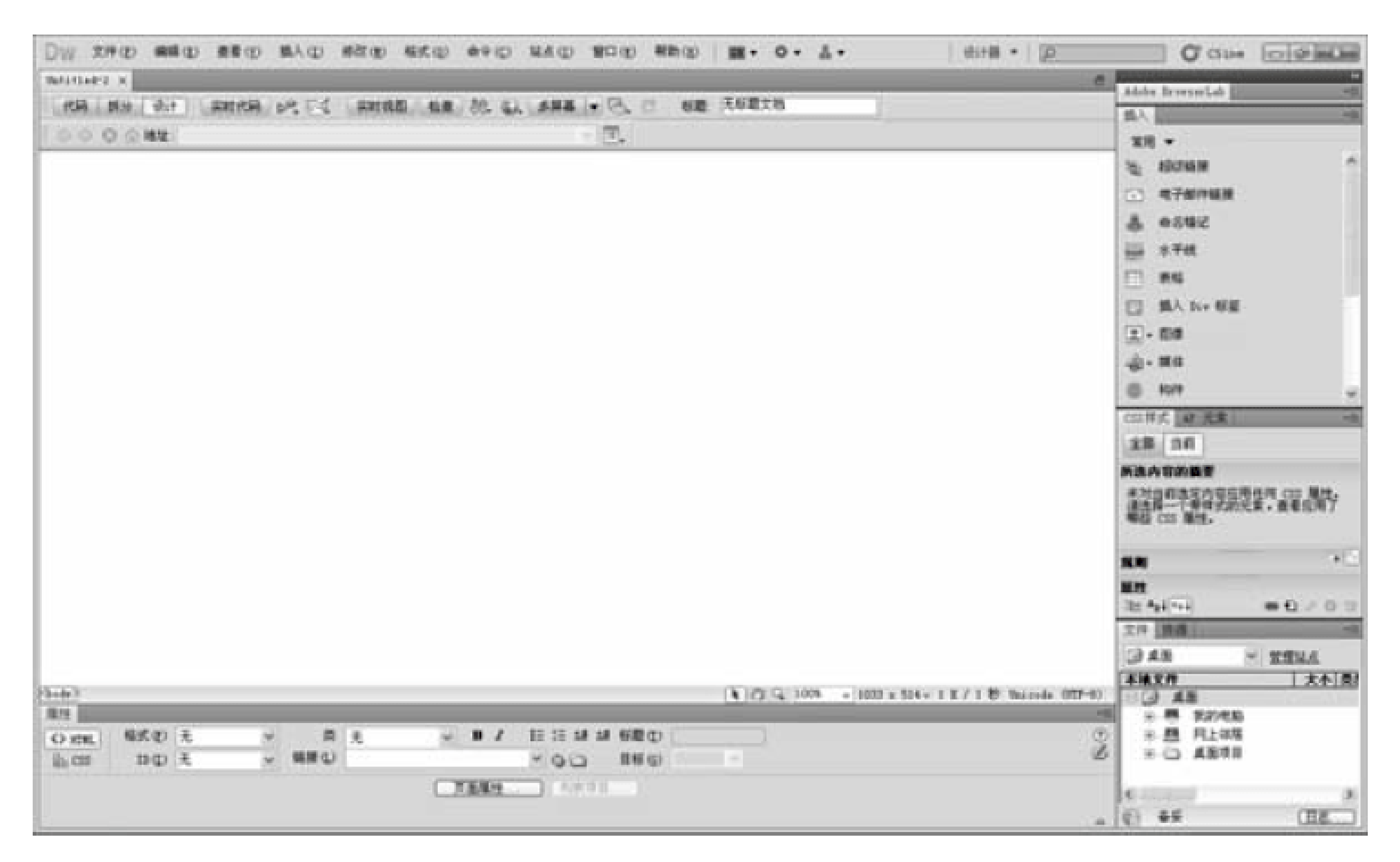


图 1-10　在设计视图下显示创建的文档

04 在文档工具栏中，单击“代码”选项卡，切换到代码视图，如图 1-11 所示。

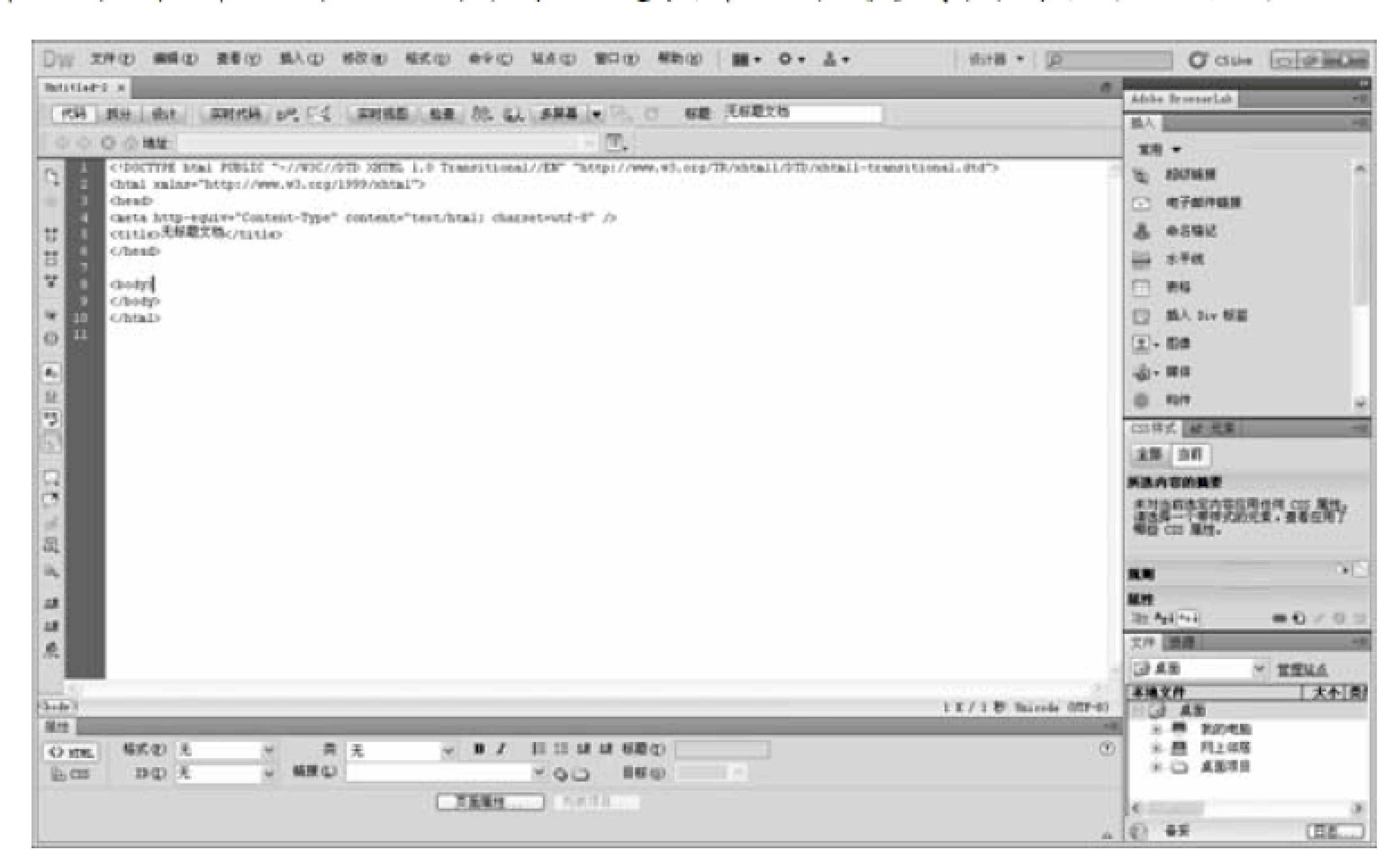


图 1-11　在代码视图下显示创建的文档

05 修改 HTML 文档标题，将代码中<title>标记中的“无标题文档”修改成“我的第一个网页”。

06 在<body>标记中键入“今天我使用 Dreamweaver CS5.5 编写了第一个简单网页，感觉非常高兴。”，完整 HTML 代码如下所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>第一个网页</title>
</head>
<body>
```

```
今天我使用 Dreamweaver CS5.5 编写了第一个简单网页，感觉非常高兴。
</body>
</html>
```

07 保存文件。单击菜单“文件”→“保存”或按下 Ctrl+S 快捷键，弹出“另存为”对话框。在对话框中选择保存位置，并输入文件名，单击“保存”按钮，如图 1-12 所示。

08 单击文档工具栏的图标，选择查看网页的浏览器，或按下功能键 F12 使用默认浏览器查看网页，预览效果如图 1-13 所示。




图 1-12　保存文件



图 1-13　浏览器预览效果

1.3　使用浏览器查看 HTML 5 文件

浏览器查看效果与 HTML 源代码是开发者经常使用的。使用浏览器查看网页的显示效果，也可以在浏览器中直接查看 HTML 源代码。

1.3.1　与 HTML 5 兼容的浏览器

浏览器是网页的运行环境，因此浏览器的种类在网页设计时一定会遇到。由于各个软件厂商对 HTML 的标准支持有所不同，导致了同样的网页在不同的浏览器下会有不同的表现。

另外，对 HTML 5 新增的功能，各个浏览器的支持程度也不一致，浏览器的因素变得比以往传统的网页设计更重要。

目前，市场上的浏览器种类繁多，Internet Explorer（以下简称 IE）是占绝对主流的，因此，本书主要使用 IE 9.0 作为主要浏览器。不过，遇到 IE 浏览器不能支持的效果，将使用 Firefox、Opera 或者其他能支持的浏览器来说明，请读者注意。

1.3.2　查看页面效果

双击前面编写的 HTML 文件，在 IE 9.0 浏览器窗口中可以看到编辑的 HTML 页面效果，请参阅图 1-3 或图 1-13。

前面已经介绍过，网页可以在不同的浏览器中查看，为了测试网页的兼容性，可以在不同

的浏览器的打开网页。

在非默认浏览器中打开网页的方法有很多种，在此为读者介绍两种常用方法。

- 方法一：选择浏览器菜单“文件”→“打开”（有些浏览的菜单项名为“打开文件”），选择要打开的网页即可。
- 方法二：在 HTML 文件上单击鼠标右键，在弹出的快捷菜单中选择“打开方式”菜单命令，单击需要的浏览器，如图 1-14 所示。如果浏览器没有出现在菜单中，选择“选择程序（C）...”选项，在计算机中查找浏览器程序。



图 1-14　选择浏览器打开网页

1.3.3　查看源文件

查看网页源代码的常见方法有以下两种。

- 在打开的页面空白处单击鼠标右键，在弹出的快捷菜中选择“查看源文件”菜单命令，如图 1-15 所示。



图 1-15　选择“查看源文件”菜单命令

- 在浏览器菜单栏上选择“查看”→“查看源文件”菜单，可以查看源文件。



图 1-16 选择“源文件”菜单命令



由于浏览器的规定各不相同，有些浏览器将“查看源文件”命名为“查看源代码”，但是操作方法完全相同。

1.4 问题解答

1．为何使用记事本编辑的 HTML 文件打开时不是在浏览器中打开，而是直接在记事本中打开？

很多初学者，保存文件时，没有将 HTML 文件的扩展名.html 或.htm 作为文件的后缀，导致文件还是以.txt 为扩展名，因此，无法在浏览器中查看。如果读者是通过单击右键，在弹出的快捷菜单中创建记事本文件的，在给文件重命名时，一定要以.html 或.htm 作为文件的后缀。特别要注意的是当 Windows 系统的扩展名是隐藏的时，更容易出现这样的错误。读者可以在“文件夹选项”对话框中查看是否显示扩展名。

2．怎样让 Dreamweaver CS6 欢迎屏幕显示与隐藏？

Dreamweaver CS5.5 欢迎屏幕可以帮助使用者快速进行打开文件、新建文件和打开相关帮助文件的操作。如果读者不希望显示该窗口，可以按下 Ctrl+U 快捷键，在弹出的窗口中选择左侧的“常规”选项，将“文档选项”部分的“显示欢迎屏幕”勾选取消即可。

第 2 章　HTML 5 中的新增元素

HTML 5 中新增了大量的元素与属性，这些新增的元素和属性使 HTML 5 的功能变得更强大，使网页设计效果有了更多的实现可能。

2.1　新增的主体结构元素

在 HTML 5 中新增了几种新的与结构相关的元素，分别是 section 元素、article 元素、aside 元素、nav 元素和 time 元素。

2.1.1　section 元素

<section>标签定义文档中的节，比如章节、页眉、页脚或文档中的其他部分。它可以与 h1、h2、h3、h4、h5、h6 等元素结合起来使用，显示文档结构。

section 标签的代码结构如下：

```
<section>
<h1>……</h1>
<p>......</p>
</section>
```

【例 2.1】（实例文件：ch02\2.1.html）

```
<!DOCTYPE HTML>
<html>
<body>
<section>
<h2>section 元素使用方法</h2>
<p>section 元素用于对网站或应用程序中页面上的内容进行分块。</p>
</section>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-1 所示，实现了内容的分块显示。



图 2-1 程序运行结果

2.1.2 article 元素

<article>标签定义外部的内容。外部内容可以是来自外部的新闻提供者的一篇新文章、来自 Blog 的文本、来自论坛的文本，或者是来自其他外部源的内容。

<article>标签的代码结构如下：

```
<article>
.....
</article>
```

【例 2.2】（实例文件：ch02\2.2.html）

```
<!DOCTYPE HTML>
<html>
<body>
<article>
  <header>
   <h1>apple 教程</h1>
   <p>时间：<time pubdate="pubdate">2013-2-1</time></p>
  </header>
  <p>轻松学习 apple 教程，就来</p>
<a href="http://www.apple.com">www.apple.com</a><br />
  <footer>
   <p><small>底部版权信息：apple.com 公司所有</small></p>
  </footer>
 </article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-2 所示，实现了外部内容的定义。



图 2-2　程序运行结果

这个实例讲述 article 元素使用方法，在 header 元素中嵌入了文章的标题部分，在标题下部的 p 元素中，嵌入了一段正文内容，在结尾处的 footer 元素中嵌入了文章的著作权作为脚注。整个示例的内容相对比较独立、完整，因此，对这部分内容使用了 article 元素。

1. article 元素与 section 元素的区别

下面再来介绍一下 article 元素与 section 元素的区别。

【例 2.3】（实例文件：ch02\2.3.html）

```
<!DOCTYPE HTML>
<html>
<body>
<article>
    <h1>article 元素与 section 元素的使用方法</h1>
    <p>何时使用 article 元素？何时使用 section 元素…..</p>
    <section>
        <h2>article 元素使用方法</h2>
        <p>article 元素代表文档、页面或应用程序中独立的、完整的、可以独自被外部引用的内容。
</p>
    </section>
    <section>
        <h2>section 元素使用方法</h2>
        <p>section 元素用于对网站或应用程序中页面上的内容进行分块。</p>
    </section>
</article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-3 所示，可以清楚地看到这两个元素的使用区别。



图 2-3　程序运行结果

2. article 元素的嵌套

article 元素是可以嵌套使用的，内层的内容在原则上需要与外层的内容相关联。例如，在一篇博客文章中，针对该文章的评论就可以使用嵌套 article 元素的方式来表示；用来呈现评论的 article 元素被包含在表示整体内容的 article 元素里面。

【例 2.4】（实例文件：ch02\2.4.html）

```
<!DOCTYPE HTML>
<html>
<body>
<article>
    <header>
        <h1>article 元素的嵌套</h1>
        <p>发表日期：<time pubdate="pubdate">2012/10/10</time></p>
    </header>
    <p>article 元素是什么？怎样使用 article 元素？……</p>
    <section>
        <h2>评论</h2>
        <article>
            <header>
                <h3>发表者：唯一</h3>
                <p><time pubdate datetime="2011-12-23T:21-26:00">1 小时前</time></p>
            </header>
            <p>这篇文章很不错啊，顶一下！</p>
        </article>
        <article>
            <header>
                <h3>发表者：唯一</h3>
                <p><time pubdate datetime="2013-2-20 T:21-26:00">1 小时前</time></p>
```

```
                </header>
                <p>这篇文章很不错啊</p>
            </article>
        </section>
</article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-4 所示。



图 2-4　程序运行结果

这个实例中的代码比较完整，它添加了读者的评论内容，实例内容分为几个部分，文章标题放在了 header 元素中，文章正文放在了 header 元素后面的 p 元素中，section 元素把正文与评论进行了区分（是一个分块元素，用来把页面中的内容进行区分），在 section 元素中嵌入了评论的内容，每一个人的评论相对又是比较独立的、完整的，因此使用一个 article 元素，在评论的 article 元素中，又可以分为标题与评论内容部分，分别放在 header 元素与 p 元素中。

2.1.3　aside 元素

aside 元素一般用来表示网站当前页面或文章的附属信息部分，它可以包含与当前页面或主要内容相关的广告、导航条、引用、侧边栏评论部分，以及其他区别于主要内容的部分。

aside 元素主要有以下两种使用方法。

第一种：被包含在 article 元素中作为主要内容的附属信息部分，其中的内容可以是与当前文章有关的相关资料、名次解释等。

aside 标签的代码结构如下所示：

```
<article>
  <h1>…</h1>
  <p>…</p>
```

```
  <aside>…</aside>
</article>
```

第二种：在 article 元素之外使用作为页面或站点全局的附属信息部分。最典型的是侧边栏，其中的内容可以是友情链接，博客中的其他文章列表、广告单元等。

aside 标签的代码结构如下所示：

```
<aside>
  <h2>…</h2>
  <ul>
     <li>…</li>
     <li>…</li>
  </ul>
  <h2>…</h2>
  <ul>
     <li>…</li>
     <li>…</li>
  </ul>
</aside>
```

【例 2.5】（实例文件：ch02\2.5.html）

```
<!DOCTYPE html>
<html>
<head>
   <title>标题文件</title>
   <link rel="stylesheet" href="mystyles.css">
</head>
<body>
   <header>
      <h1>站点主标题</h1>
   </header>
   <nav>
      <ul>
         <li>主页</li>
         <li>图片</li>
         <li>音频</li>
      </ul>
   </nav>
   <section>
  </section>
   <aside>
      <blockquote>文章 1</blockquote>
```

```
        <blockquote>文章 2</blockquote>
    </aside>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-5 所示。



图 2-5　程序运行结果



<aside>元素可以位于示例页面的左边或右边，并没有预定义的位置，可位于布局的任意部分，用于表示任何非文档主要内容的部分。例如，可以在<section>元素中加入<aside>元素，甚至可以把该元素加入一些重要信息，如文字引用。<aside>元素仅仅描述所包含的信息，而不反映结构。

2.1.4　nav 元素

<nav>用来将具有导航性质的链接划分在一起，使代码结构在语义化方面更加准确，同时对于屏幕阅读器等设备的支持也更好。

具体来说，nav 元素可以用于以下场合。

- 传统导航条：现在主流网站上都有不同层级的导航条，作用是将当前画面跳转到的其他主页上。
- 侧边栏导航：现在主流博客网站及商品网站上都有侧边栏导航，其作用是将页面从当前文章或当前商品跳转到其他文章或其他商品页上。
- 页内导航：页内导航的作用是在本页的组成部分之间进行跳转。
- 翻页操作：翻页操作是指在网站的多个页面的前后页面或博客网站的前后篇文章之间滚动。

具体实现代码如下：

```
<nav>
```

```
<a href="……">Home</a>
<a href="……">Previous</a>
<a href="……">Next</a>
</nav>
```



如果文档中有“前后”按钮，则应该把它放到<nav>元素中。

一个页面中可以拥有多个<nav>元素，作为页面整体或不同部分的导航；下面给出一个代码实例。

【例 2.6】（实例文件：ch02\2.6.html）

```
<!DOCTYPE html>
<html>
<body>
<h1>技术资料</h1>
<nav>
    <ul>
        <li><a href="/">主页</a></li>
        <li><a href="/events">开发文档</a></li>
    </ul>
</nav>
<article>
  <header>
        <h1>HTML 5 与 CSS 3 的历史</h1>
        <nav>
            <ul>
                <li><a href="#HTML 5">HTML 5 的历史</a></li>
                <li><a href="#CSS 3">CSS 3 的历史</a></li>
            </ul>
        </nav>
  </header>
  <section id="HTML 5">
        <h1>HTML 5 的历史</h1>
        <p>讲述 HTML 5 的历史的正文</p>
        <footer>
        <p>
            <a href="?edit">已往版本</a>|
            <a href="?delete">当前现状</a>|
            <a href="?rename">未来前景</a>
        </p>
  </footer>
```

```
    </section>
    <section id="CSS 3">
        <h1>CSS 3 的历史</h1>
        <p>讲述 CSS 3 的历史的正文</p>
    </section>
    <footer>
        <p>
            <a href="?edit">已往版本</a>|
            <a href="?delete">当前现状</a>|
            <a href="?rename">未来前景</a>
        </p>
    </footer>
</article>
<footer>
   <p><small>版权所有：青花瓷</small></p>
</footer>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-6 所示。

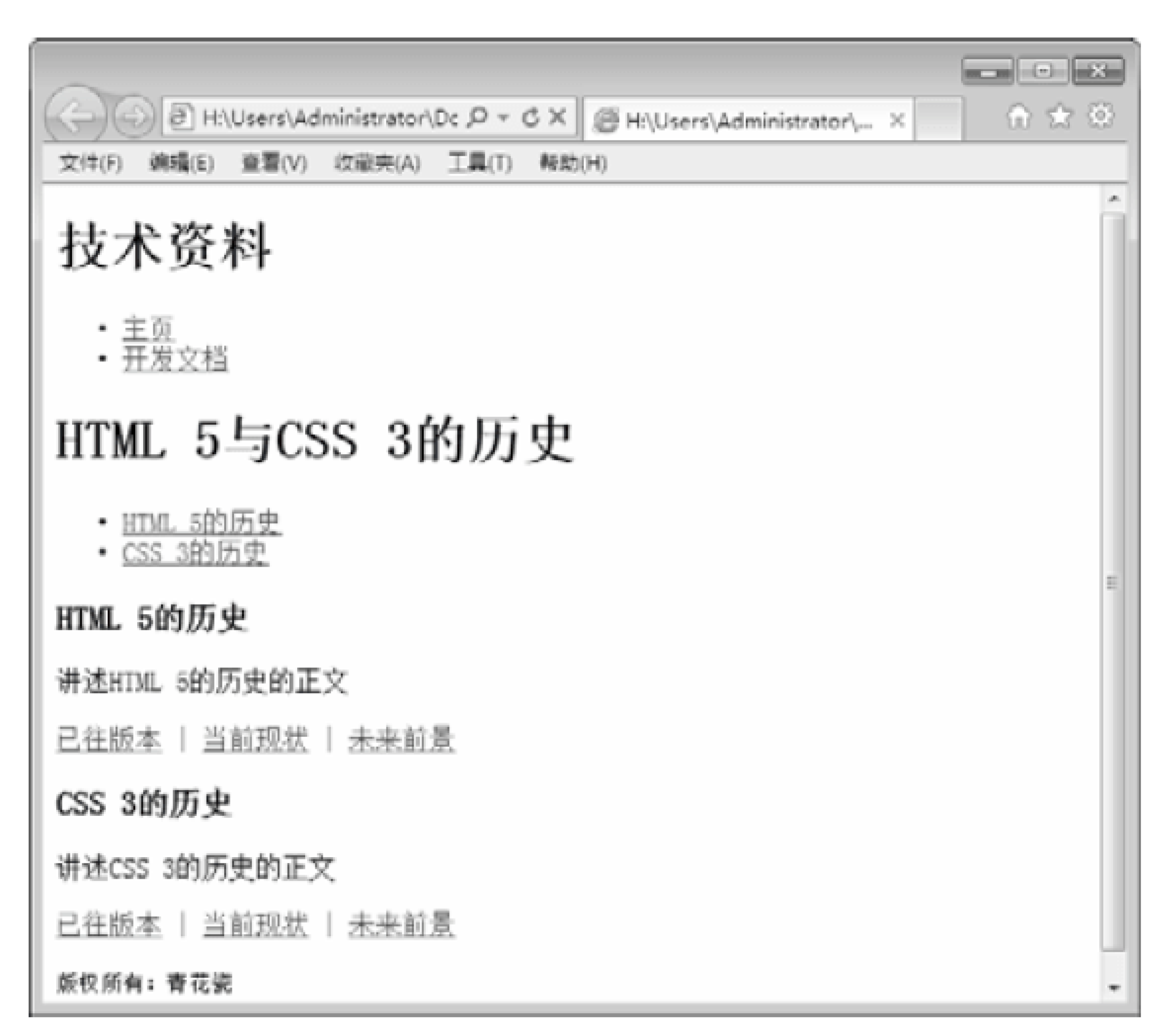


图 2-6　程序运行结果



提 示

在这个实例中，可以看到<nav>不仅可以作为页面的全局导航，也可以放在<article>标签内，作为单篇文章内容的相关导航链接到当前页面的其他位置。



注 意

在 HTML 5 中不要用 menu 元素代替 nav 元素，menu 元素是用在一系列发出命令的菜单上的，是一种交互性的元素，或者更确切地说是使用在 Web 应用程序中的。

2.1.5 time 元素

<time>是 HTML 5 新增加的一个标记，用于定义时间或日期。该元素可以代表 24 小时中的某一时刻，在表示时刻时，允许有时间差。在设置时间或日期时，只需将该元素的属性 datetime 设为相应的时间或日期即可。

具体实现代码如下：

```
<p>
<time>
……
</time>。
</p>
<p>
<time datetime=
……
</time>
</p>
```

【例 2.7】（实例文件：ch02\2.7.html）

```
<!DOCTYPE html>
<html>
<body>
<h1>Time 元素</h1>
<p id="p1">
 <time datetime="2013-3-17">
今天是 2013 年 3 月 17 日
 </time>
<p>
<p id="p2">
 <time datetime="2013-3-17T17:00">
现在时间是 2013 年 3 月 17 日晚上 5 点
 </time>
<p>
<p id="p3">
  <time datetime="2013-12-31">
    新款冬装将于今年年底上市
  </time>
</p>
 <p id="p4">
   <time datetime="2013-3-15" pubdate="true">
    本消息发布于 2013 年 3 月 15 日
   </time>
```

```
</p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-7 所示。



图 2-7 程序运行结果

说明：

- <p>元素中 ID 号为 p1 的<time>元素表示的是日期。页面在解析时，获取的是属性 datetime 中的值，而标记之间的内容只显示在页面中。
- <p>元素中 ID 号为 p2 的<time>元素表示的是日期和时间，使用字母 T 进行分隔。如果在整个日期与时间后面加字母 Z，则表示获取的是 UTC（世界统一时间）格式。
- <p>元素中 ID 号为 p3 的<time>元素表示的是将来时间。
- <p>元素中 ID 号为 p4 的<time>元素表示的是发布日期。



为了在文档中将这两个日期进行区分，在最后一个<time>元素中增加了 pubdate 属性，表示此日期为发布日期。



<time>元素中的可选属性 pubdate 表示时间是否为发布日期，是一个布尔值，该属性不仅可以用于<time>元素，还可用于<article>元素。

2.2 新增的非主体结构元素

在 HTML 5 中还新增了一些非主体的结构元素，如：header、hgroup、footer 等。

2.2.1 header 元素

header 元素是一种具有引导和导航作用的结构元素，通常用来放置整个页面或页面内的内容区块的标题，但也可以包含其他内容，例如数据表格、搜索表单或相关的 logo 图片。

header 标签的代码结构如下：

```
<header>
<h1>……</h1>
<p>……</p>
</header>
```

在整个页面中标题一般放在页面的开头，一个网页中没有限制 header 元素的个数，可以拥有多个，可以为每个内容区块加一个 header 元素。

【例 2.8】（实例文件：ch02\2.8.html）

```
<!DOCTYPE html>
<html>
<body>
<header>
 <h1>网页标题</h1>
</header>
<article>
 <header>
    <h1>文章标题</h1>
 </header>
 <p>文章正文</p>
</article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-8 所示。

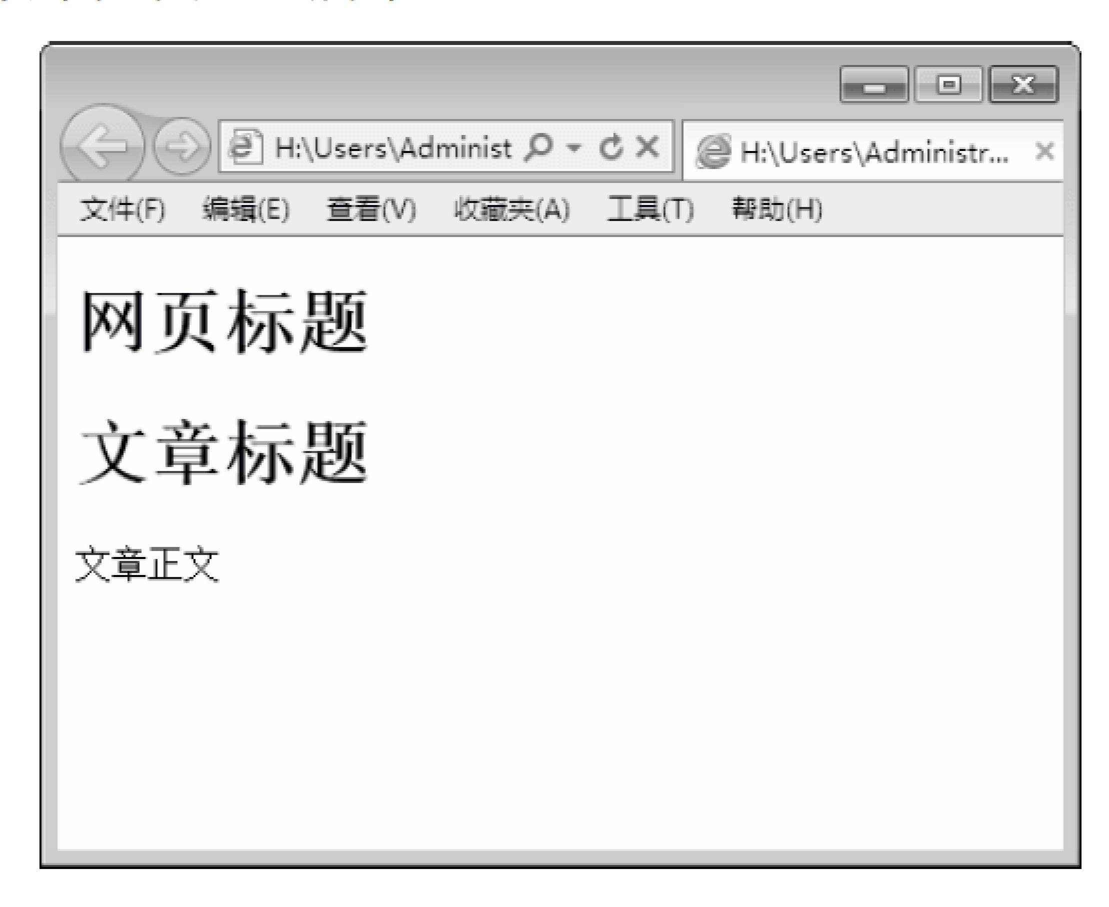


图 2-8　程序运行结果



在 HTML 5 中，一个 header 元素通常至少包括一个 headering 元素（h1~h6），也可以包括 hgroup 元素、nav 元素，还可以包括其他元素。

2.2.2 hgroup 元素

<hgroup>标签用于对网页或区段（section）的标题进行组合。hgroup 元素通常会将 h1～h6 元素进行分组，譬如将一个内容区块的标题及其子标题分为一组。

hgroup 标签的使用代码如下：

```
<hgroup>
 <h1>……</h1>
 <h2>……t</h2>
</hgroup>
```

通常，如果文章只有一个主标题，是不需要 hgroup 元素的。如下这个实例就不需要 hgroup 元素。

【例 2.9】（实例文件：ch02\2.9.html）

```
<!DOCTYPE html>
<html>
<body>
<article>
    <header>
        <h1>文章标题</h1>
        <p><time datetime="2010-03-20">2010 年 10 月 29 日</time></p>
    </header>
    <p>文章正文</p>
</article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-9 所示。

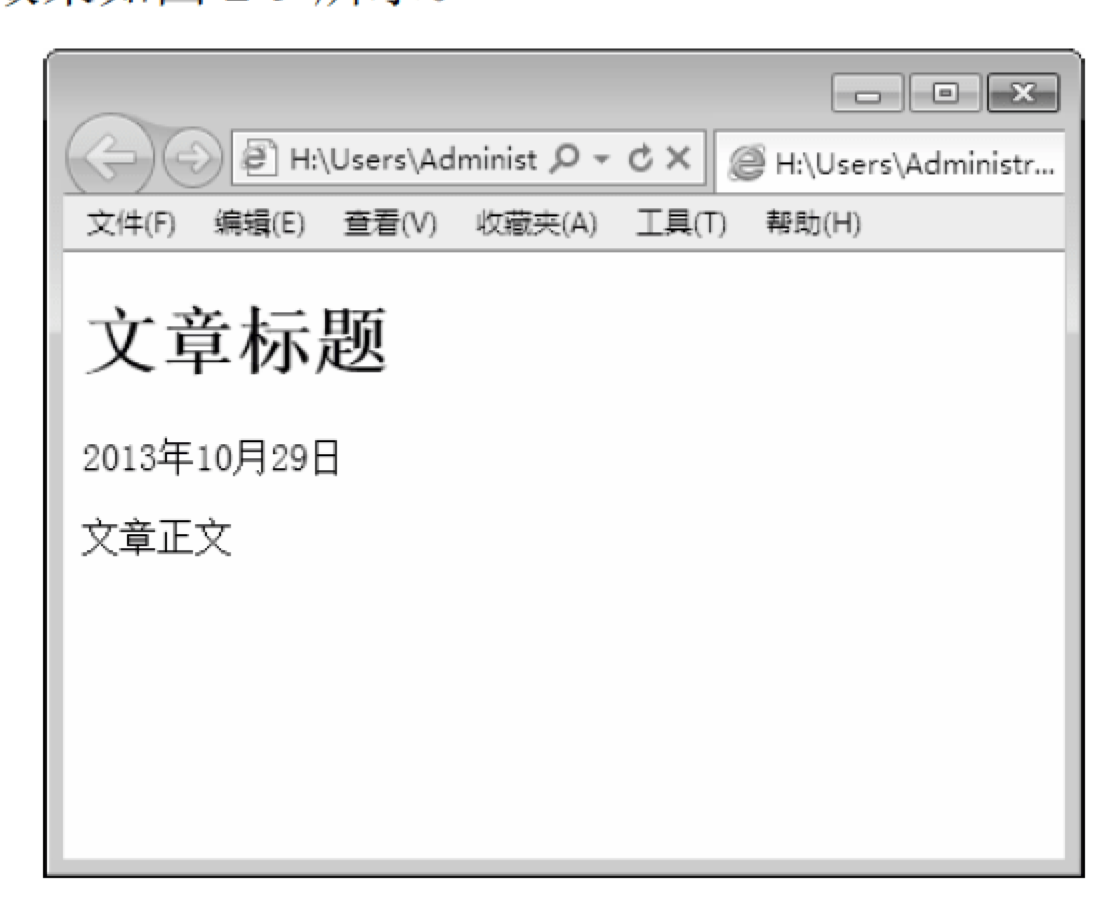


图 2-9　程序运行结果

但是，如果文章有主标题和子标题，就需要使用 hgroup 元素。如下这个实例就需要 hgroup

元素。

【例 2.10】（实例文件：ch02\2.10.html）

```
<!DOCTYPE html>
<html>
<body>
<article>
    <header>
        <hgroup>
            <h1>文章主标题</h1>
            <h2>文章子标题</h2>
        </hgroup>
        <p><time datetime="2013-03-20">2013 年 10 月 29 日</time></p>
    </header>
    <p>文章正文</p>
</article>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-10 所示。

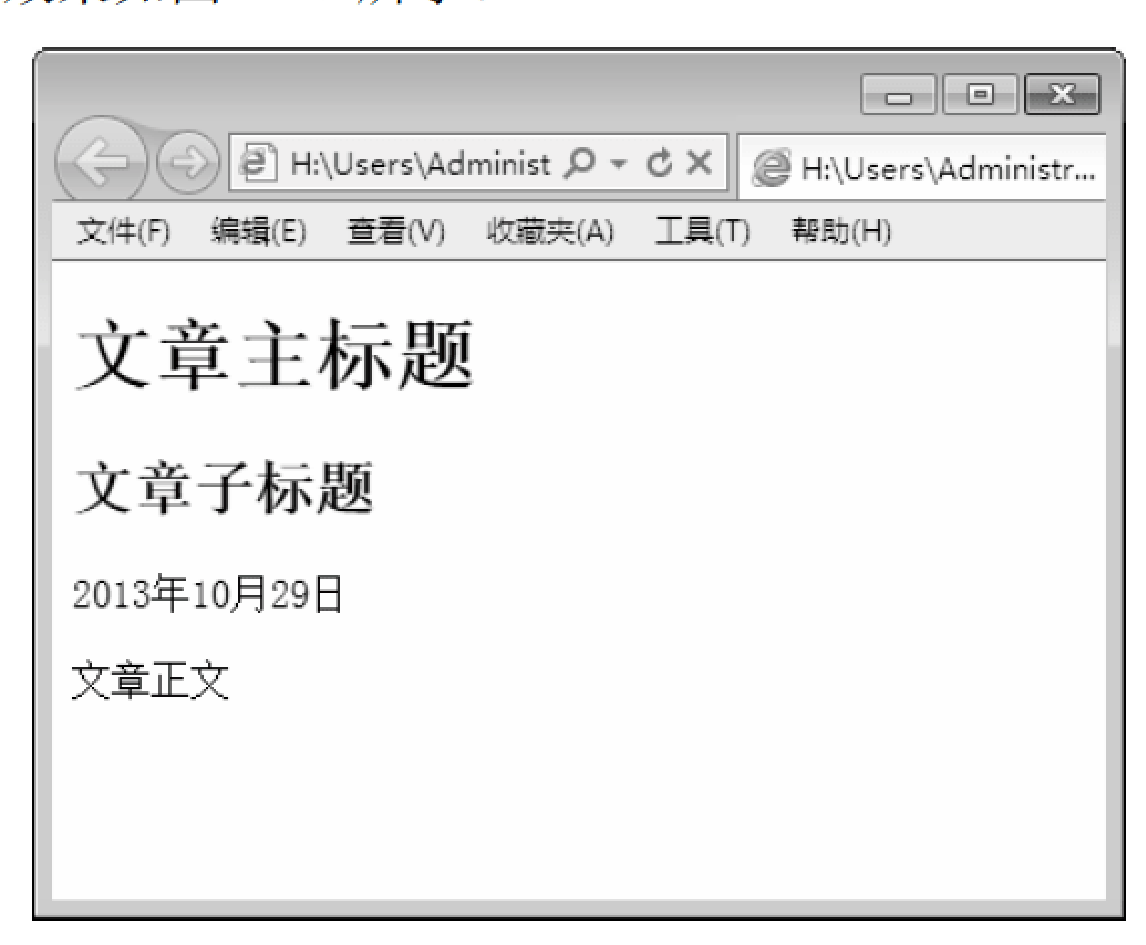


图 2-10 程序运行结果

2.2.3 footer 元素

footer 元素可以作为其上层父级内容区块或是一个根区块的脚注。footer 通常包括其相关区块的脚注信息，如作者、相关阅读链接及版权信息等。

使用 footer 标签设置文档页脚的代码如下：

```
<footer>……</footer>
```

在 HTML 5 出现之前，网页设计人员使用下面的方式编写页脚。

【例 2.11】（实例文件：ch02\2.11.html）

```
<!DOCTYPE html>
<html>
<body>
<div id="footer">
    <ul>
        <li>版权信息</li>
        <li>站点地图</li>
        <li>联系方式</li>
    </ul>
<div>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-11 所示。

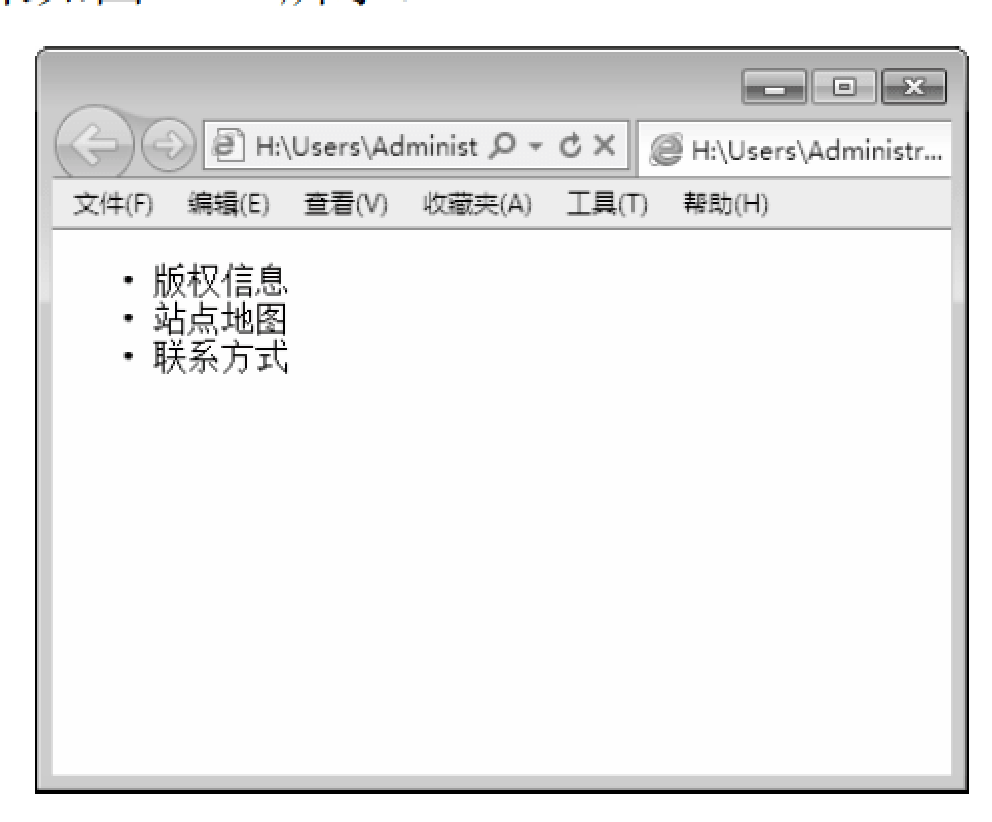


图 2-11　程序运行结果

但是到了 HTML 5 之后，这种方式将不再使用，而是使用更加语义化的 footer 元素来替代。

【例 2.12】（实例文件：ch02\2.12.html）

```
<!DOCTYPE html>
<html>
<body>
<footer>
    <ul>
        <li>版权信息</li>
        <li>站点地图</li>
        <li>联系方式</li>
    </ul>
</footer>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-12 所示。

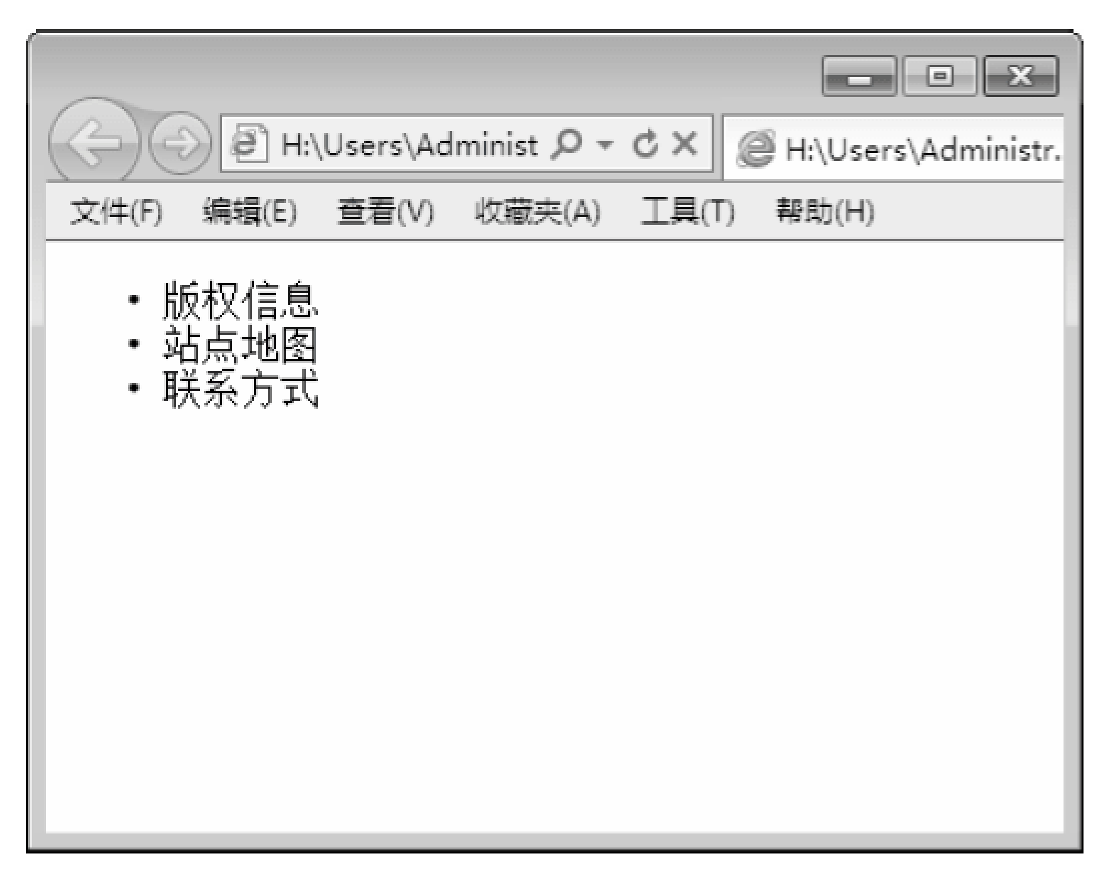


图 2-12　程序运行结果



与 header 元素一样，页面中也未限制 footer 元素的个数。同时，可以为 article 元素或 section 元素添加 footer 元素。

【例 2.13】（实例文件：ch02\2.13.html）

```
<!DOCTYPE html>
<html>
<body>
<article>
        文章内容
        <footer>
              文章的脚注
        </footer>
</article>
<section>
        分段内容
        <footer>
              分段内容的脚注
        </footer>
</section>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-13 所示。

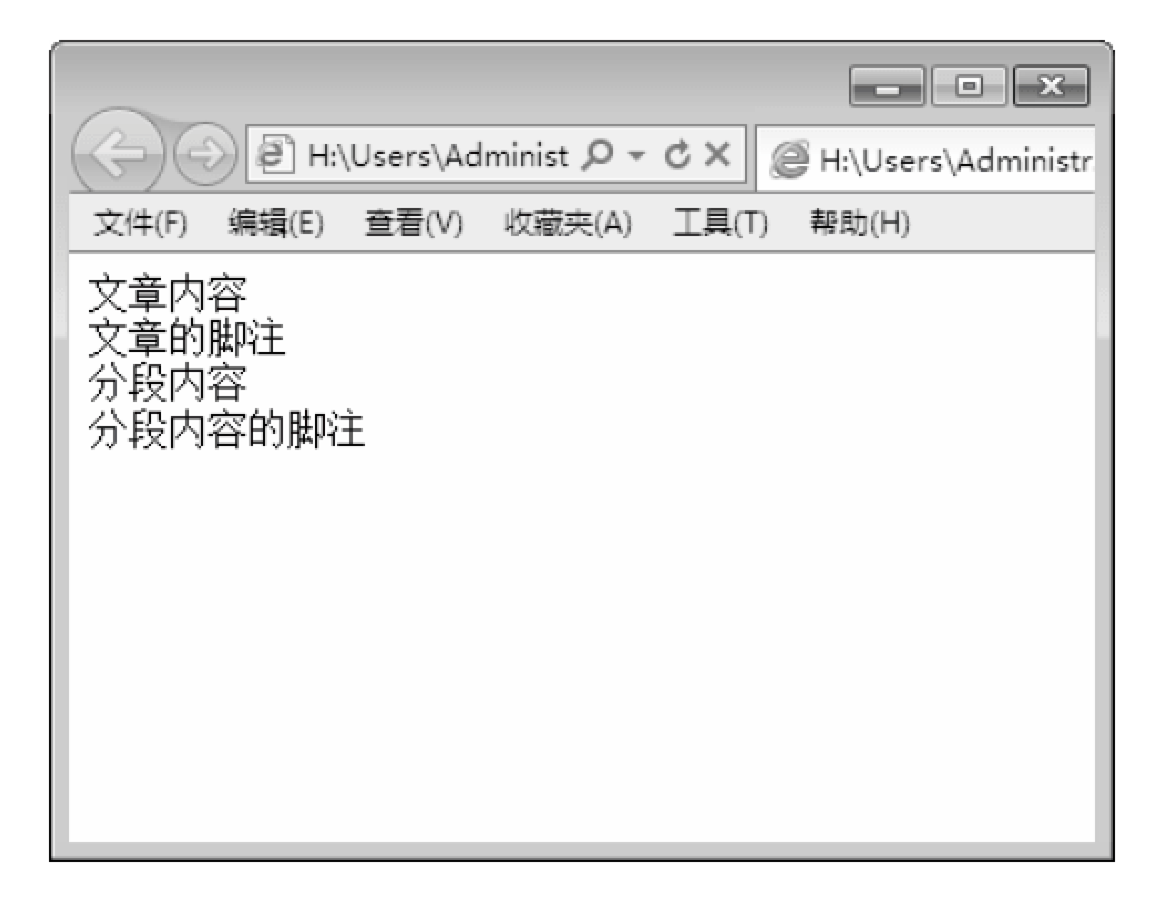


图 2-13 程序运行结果

2.2.4 figure 元素

figure 元素是一种元素的组合，可带有标题（可选）。figure 元素用来表示网页上一块独立的内容，将其从网页上移除后不会对网页上的其他内容产生影响。figure 所表示的内容可以是图片、统计图或代码示例。

figure 元素的实现代码如下：

```
<figure>
<h1>……</h1>
<p>……</p>
</figure>
```



使用 figure 元素时，需要用 figcaption 元素为 figure 元素组添加标题。不过，figure 元素内最多只允许放置一个 figcaption 元素，其他元素可无限放置。

1. 不带标题的 figure 元素的使用

【例 2.14】不带标题的 figure 元素的使用（实例文件：ch02\2.14.html）

```
<!DOCTYPE HTML>
<html>
<head>
<title>不带有标题的 figure 元素</title>
</head>
<body>
    <figure>
        <img alt="images/logo.jpg"/>
    </figure>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-14 所示。

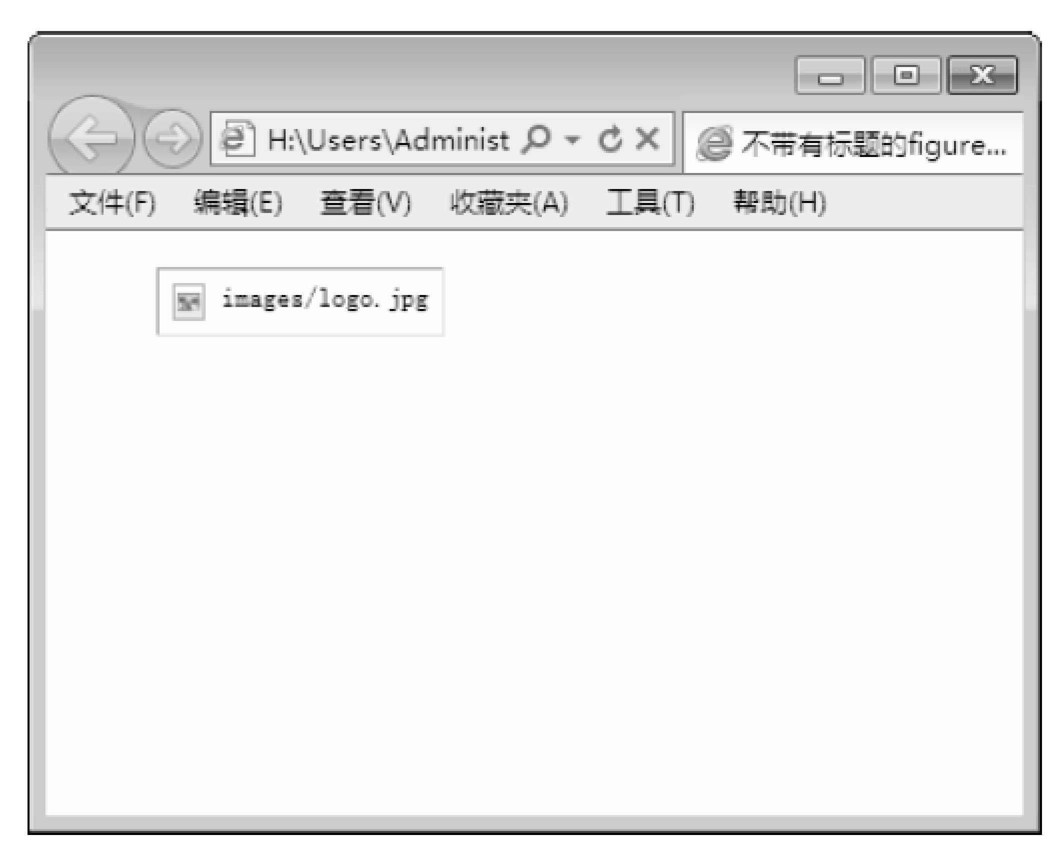


图 2-14　程序运行结果

2. 带有标题的 figure 元素的使用

【例 2.15】带有标题的 figure 元素的使用（实例文件：ch02\2.15.html）

```
<!DOCTYPE HTML>
<html>
<head>
<title>带有标题的 figure 元素</title>
</head>
<body>
    <figure>
        <img alt="images/logo.jpg"/>
    </figure>
<figcaption>标题提示</figcaption>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-15 所示。

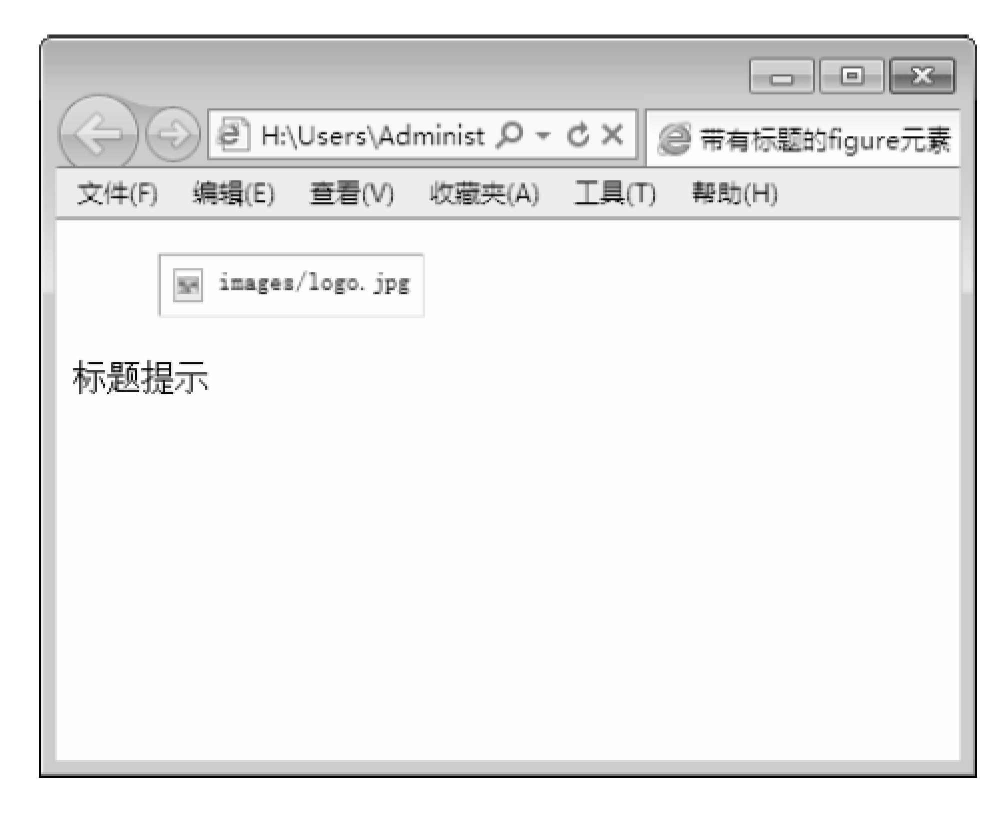


图 2-15　程序运行结果

3. 多张图片用同一标题的 figure 元素的使用

【例 2.16】多张图片用同一标题的 figure 元素的使用（实例文件：ch02\2.16.html）

```
<!DOCTYPE HTML>
<html>
<head>
<title>多张图片，同一标题的 figure 元素</title>
</head>
<body>
    <figure>
        <img alt="images/logo.jpg"/>
<img alt="images/logo1.jpg"/>
<img alt="images/logo2.jpg"/>
    </figure>
<figcaption>标题提示</figcaption>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-16 所示。

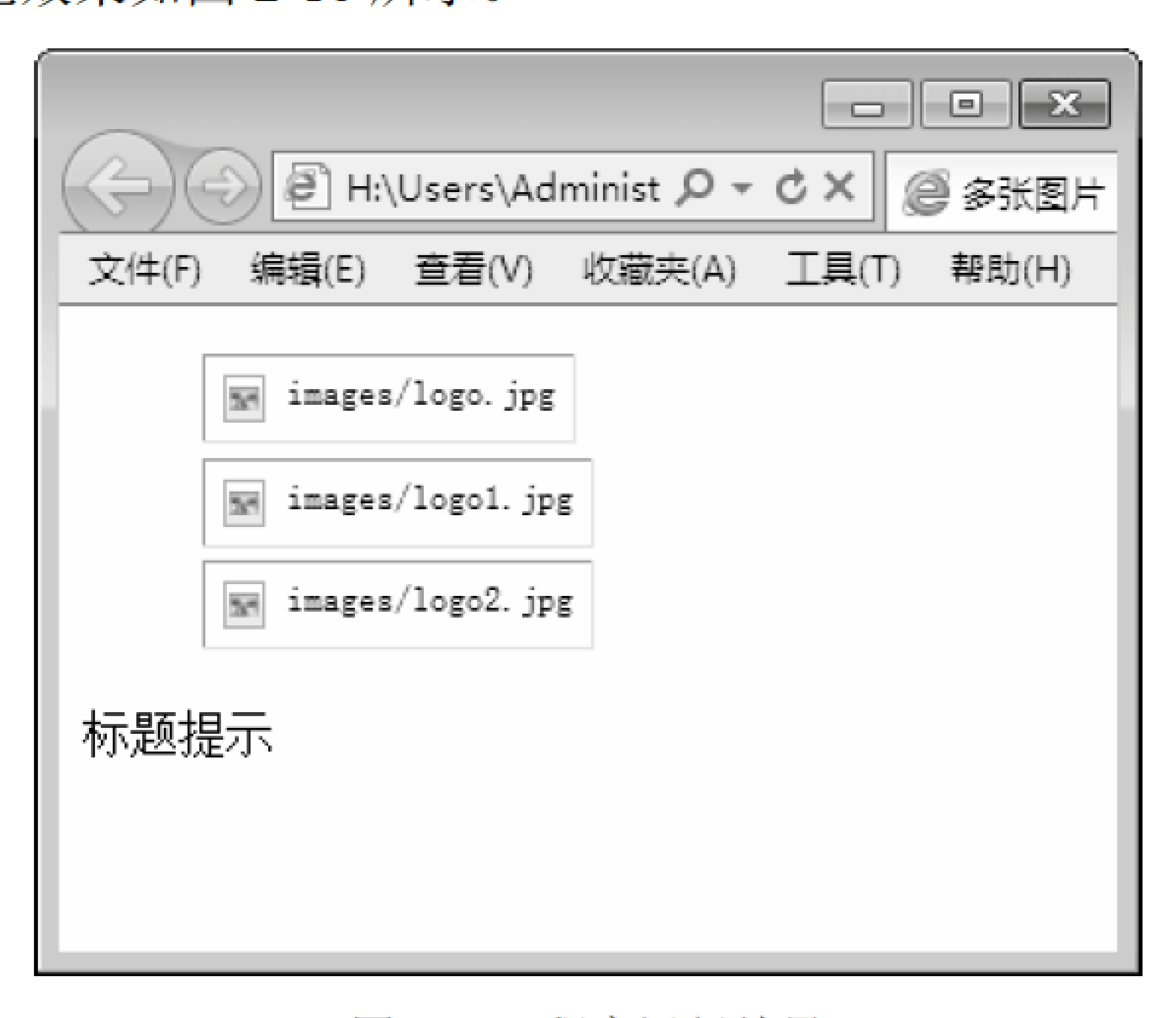


图 2-16　程序运行结果

2.2.5　address 元素

address 元素用来在文档中呈现联系信息，包括文档作者或文档维护者姓名、网站链接地址、电子邮箱、真实地址和电话号码等。

address 元素的实现代码如下：

```
<address>
        <a href=……>……</a>
……
```

```
</address>
```

【例 2.17】address 元素的使用（实例文件：ch02\2.17.html）

```
<!DOCTYPE html>
<html>
<body>
<address>
        <a href=http://blog.sina.com.cn/zhangsan>张三</a>
       <a href=http://blog.sina.com.cn/lisi>李四</a>
       <a href=http://blog.sina.com.cn/wanger>王二</a>
</address>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-17 所示。

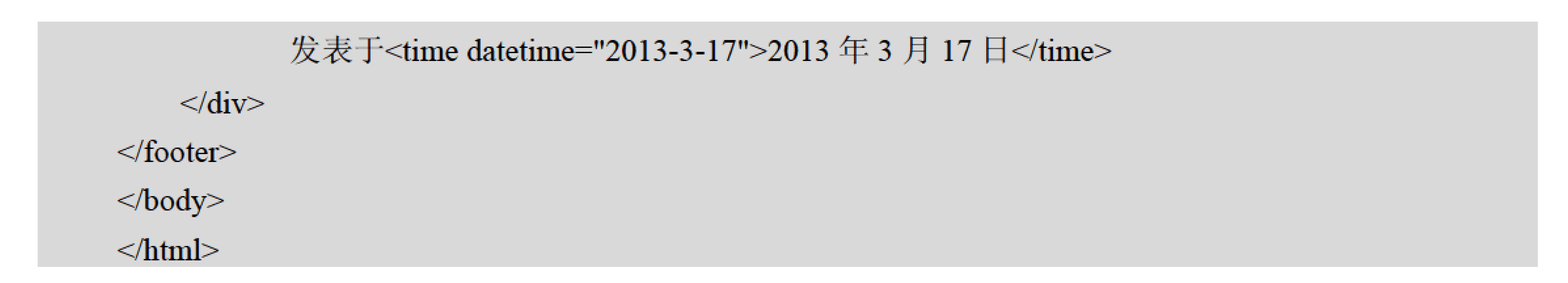


图 2-17　程序运行结果

另外，address 元素不仅可以单独使用，还可以与 footer 元素、time 元素与 address 元素结合起来使用。

【例 2.18】address 元素与其他元素结合使用（实例文件：ch02\2.18.html）

```
<!DOCTYPE html>
<html>
<body>
<footer>
    <div>
        <address>
           <a title="文章作者：张三" href="http://blog.sina.com.cn/zhangsan">
            张三</a>
        </address>
```



```
            发表于<time datetime="2013-3-17">2013 年 3 月 17 日</time>
    </div>
</footer>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-18 所示。

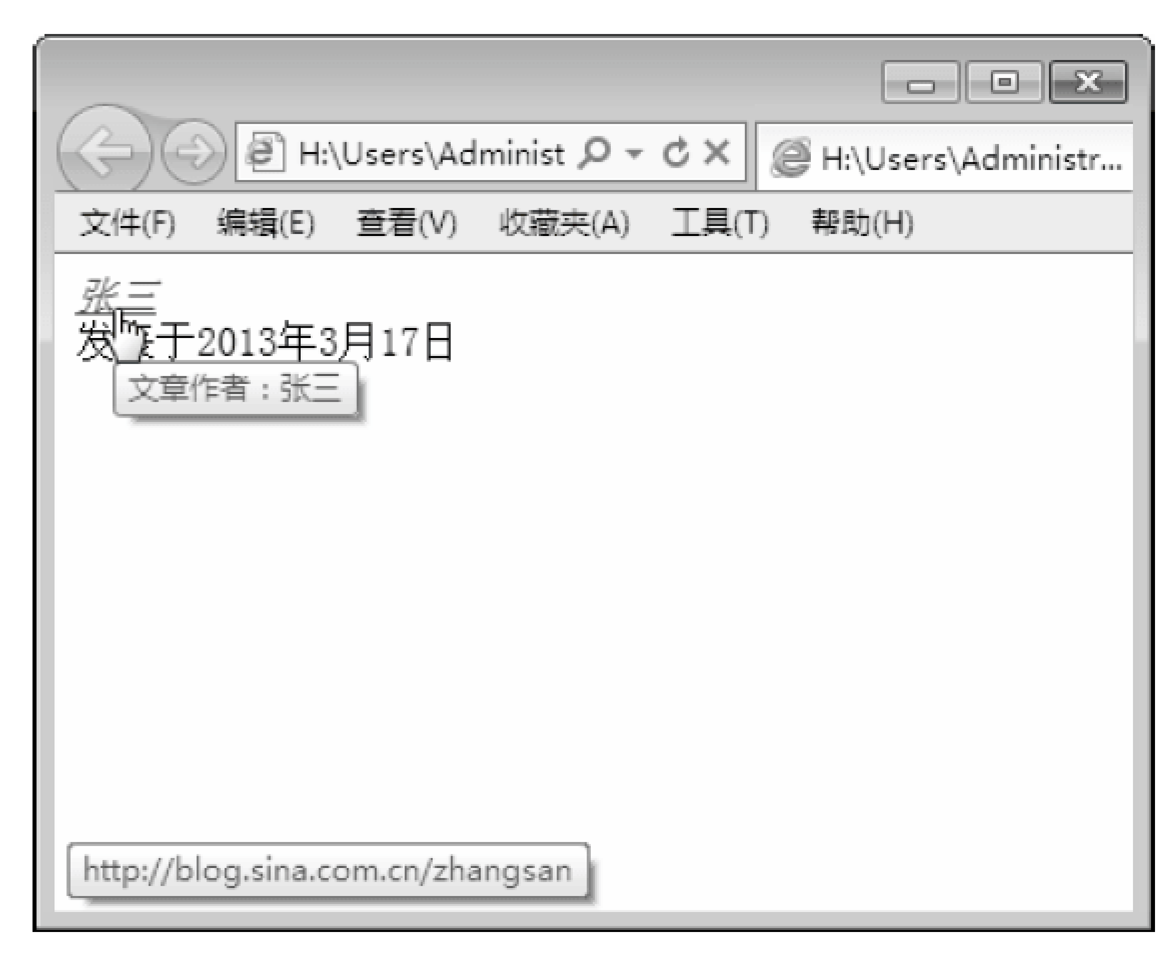


图 2-18　程序运行结果

2.3　新增其他常用元素

除了结构元素外，在 HTML 5 中，还新增了其他元素，如 progress 元素、command 元素、embed 元素、mark 元素和 details 元素等。

2.3.1　mark 元素

mark 元素主要用来呈现需要突出显示或高亮显示的文字。mark 元素的典型应用是在搜索结果中向用户高亮显示搜索关键词。其使用方法与<em>和<strong>有相似之处，但相比而言，HTML 5 中新增的<mark>元素在突出显示时，更加随意与灵活。

HTML 5 中代码示例：

```
p>……<mark>……</mark>……</p>
```

【例 2.19】mark 元素的使用（实例文件：ch02\2.19.html）

在页面中，首先使用<h5>元素创建一个标题“优秀开发人员的素质”，然后通过<p>元素对标题进行阐述。在阐述的文字中，为了引起用户的注意，使用<mark>元素高亮处理字符“素质”、“过硬”和“务实”等字样。

具体的代码如下：

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="utf-8" />
<title>mark 元素的使用</title>
<link href="Css/css3.css" rel="stylesheet" type="text/css">
</head>
<body>
 <h5>优秀开发人员的<mark>素质</mark></h5>
 <p class="p3_5">
      一个优秀的 Web 页面开发人员，必须具有
   <mark>过硬</mark>的技术与
   <mark>务实</mark>的专业精神
</p>
</body>
</html>
```

该页面在 IE 9.0 浏览器下执行的效果如图 2-19 所示。

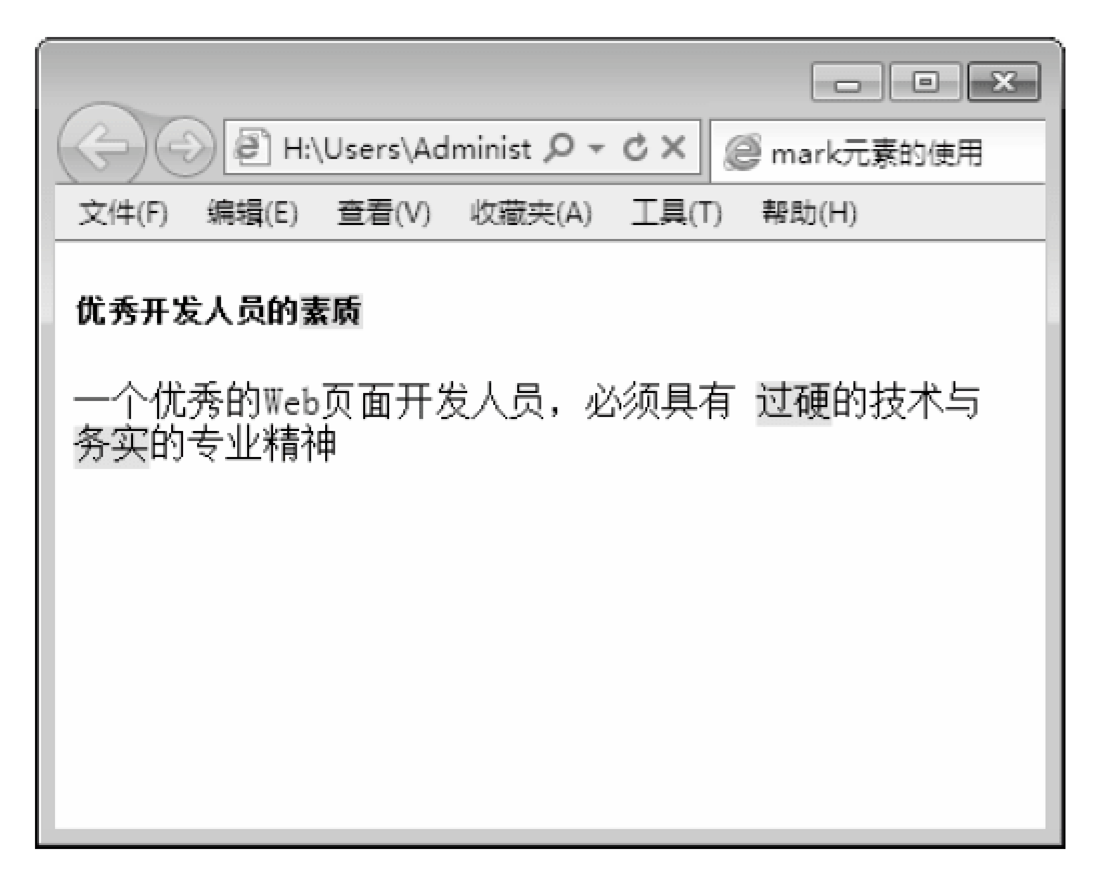


图 2-19　程序运行结果



<mark>元素的这种高亮显示的特征，除在文档中突出显示外，还常用于查看搜索结果页面中关键字的高亮显示，其目的主要是引起用户的注意。



虽然<mark>元素在使用效果上与<em>或<strong>元素有相似之处，但三者的出发点是不一样的。<strong>元素是作者对文档中某段文字的重要性进行的强调；<em>元素是作者为了突出文章重点而进行的设置；<mark>元素是在数据展示时，以高亮形式显示某些字符，与原作者本意无关。

2.3.2　rp、rt 与 ruby 元素

ruby 元素由一个或多个字符（需要一个解释/发音）和一个提供该信息的 rt 元素组成，还

包括可选的 rp 元素，定义当浏览器不支持 ruby 元素时显示的内容。

rp、rt 与 ruby 元素结合使用的代码如下：

```
<ruby>
<rt><rp>(</rp><rp>)</rp></rt>
</ruby>
```

【例 2.20】使用 ruby 注释繁体字“漢”（实例文件：ch02\2.20.html）

```
<!DOCTYPE html>
<html>
<body>
<ruby>
漢<rt><rp>(</rp>汉<rp>)</rp></rt>
</ruby>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-20 所示。

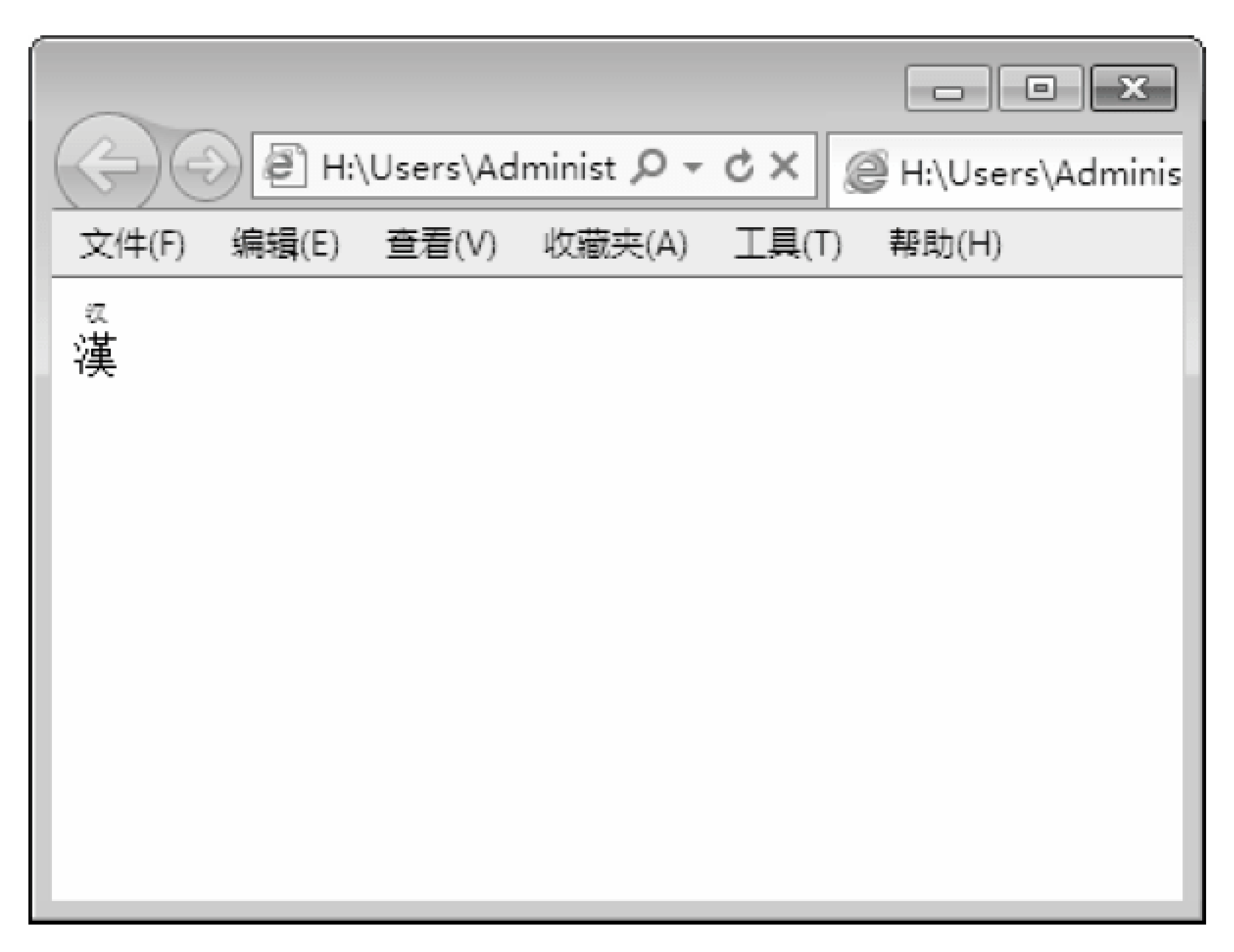


图 2-20　程序运行结果



支持 ruby 元素的浏览器不会显示 rp 元素的内容。

2.3.3　progress 元素

progress 元素表示运行中的进程，可以使用 progress 元素来显示 JavaScript 中耗费时间的函数进程。例如下载文件时，文件下载到本地的进度值，可以通过该元素动态展示在页面中，展示的方式既可以使用整数（如 1～100），也可以使用百分比（如 10%～100%）。

<progress>元素的属性及描述如下表所示。

属性	值	描述
max	整数或浮点数	设置完成时的值，表示总体工作量
value	整数或浮点数	设置正在进行时的值，表示已完成的工作量



<progress>元素中设置的 value 值必须小于或等于 max 属性值，且两者都必须大于 0。

【例 2.21】使用 progress 元素表示下载进度（实例文件：ch02\2.21.html）

```
<!DOCTYPE HTML>
<html>
<body>
  对象的下载进度：
  <progress>
   <span id="objprogress">76</span>%
  </progress>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-21 所示。

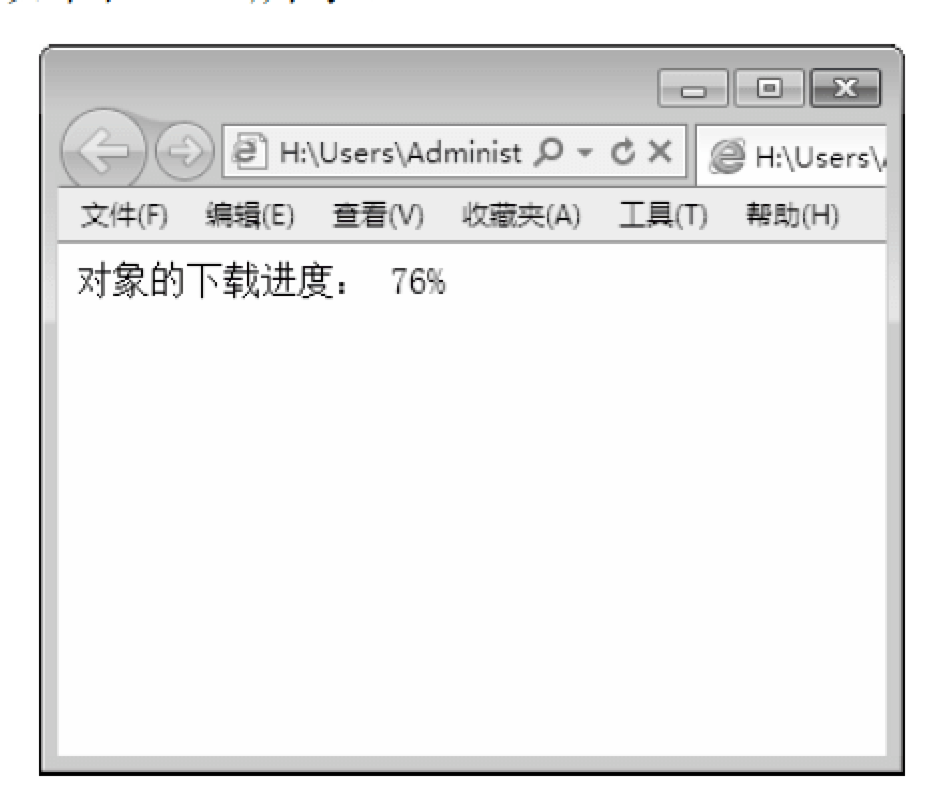


图 2-21　程序运行结果

2.3.4　command 元素

command 元素表示用户能够调用的命令，可以定义命令按钮，比如单选按钮、复选框或按钮。

HTML 5 中使用的代码：

```
<command type="command">……</command>
```

【例 2.22】使用 command 元素标记一个按钮（实例文件：ch02\2.22.html）

```
<!DOCTYPE HTML>
```

```
<html>
<body>
  <menu>
    <command onclick="alert('Hello World')">Click Me!</command>
  </menu>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-22 所示。单击网页中的“Click Me”区域，将弹出提示信息框。

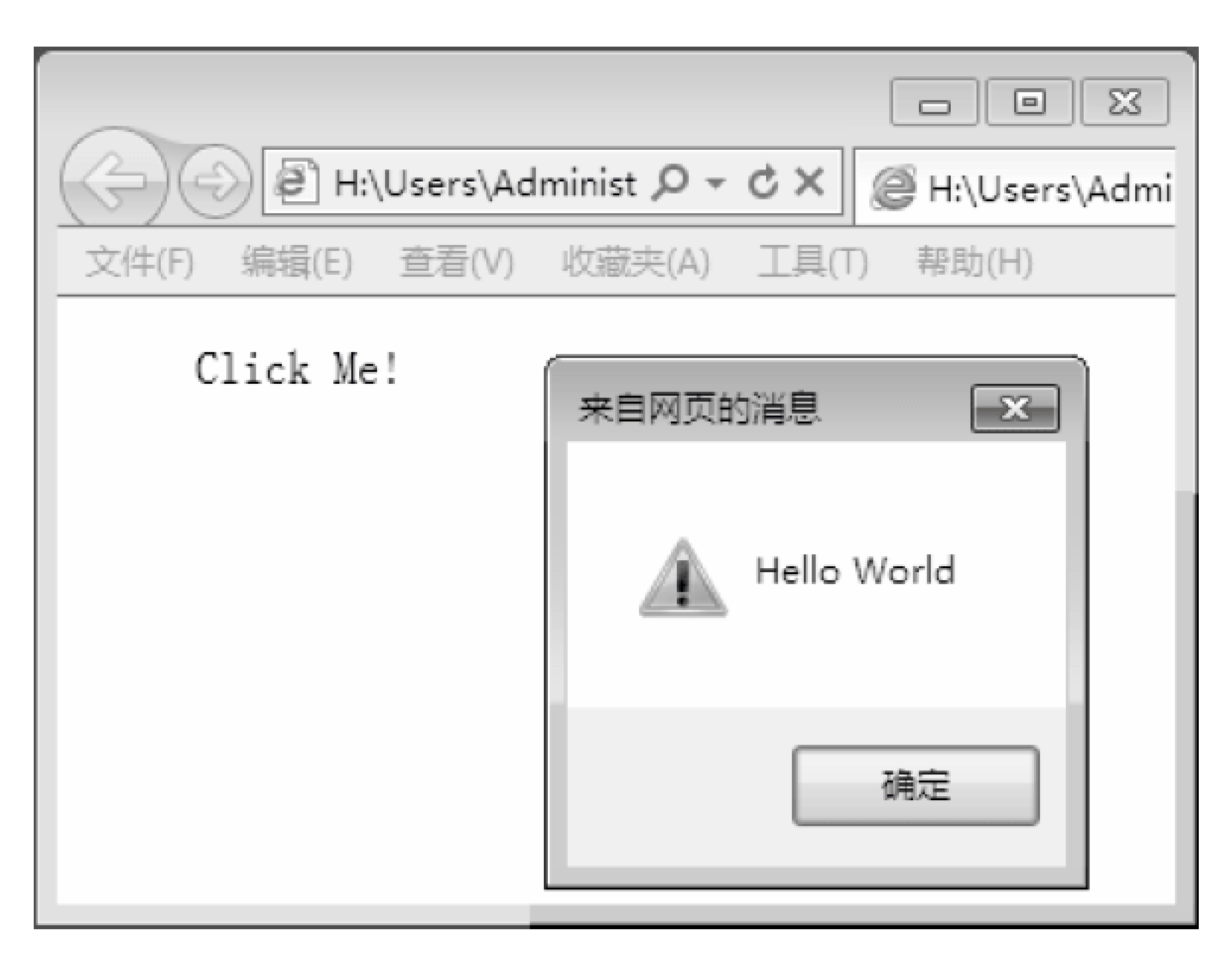


图 2-22　程序运行结果



只有当 command 元素位于 menu 元素内时，该元素才是可见的。否则不会显示这个元素，但是可以用它规定键盘快捷键。

2.3.5　embed 元素

embed 元素用来插入各种多媒体，格式可以是 Midi、Wav、AIFF、AU、MP3 等。

HTML 5 中代码示例：

```
<embed src="……"/>
```

【例 2.23】使用 embed 元素插入动画（实例文件：ch02\2.23.html）

```
<!DOCTYPE HTML>
<html>
<body>
<embed src="images/飞翔的海鸟.swf"/>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-23 所示。

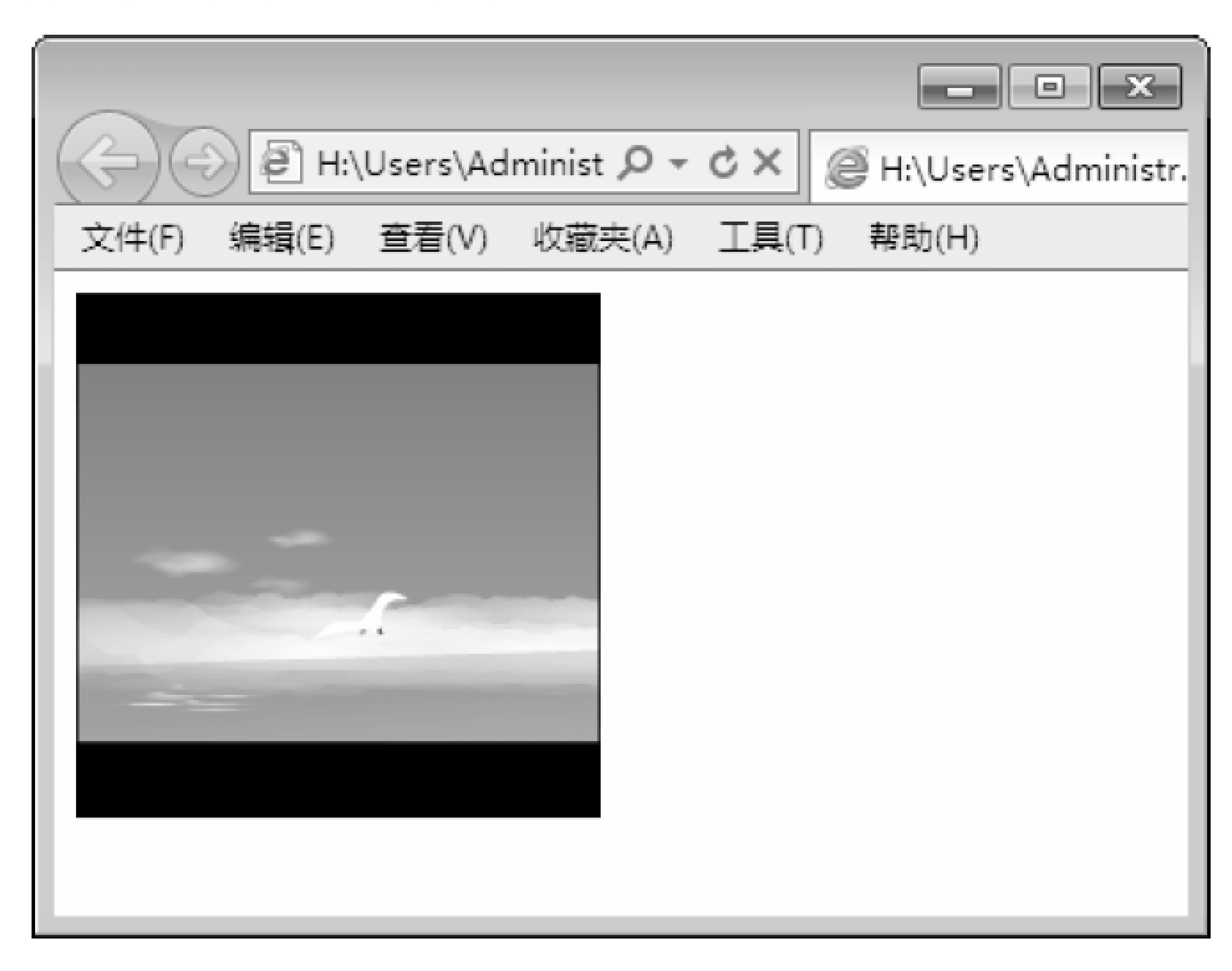


图 2-23　程序运行结果

2.3.6　details 与 summary 元素

details 元素表示用户要求得到并且可以得到的细节信息，与 summary 元素配合使用。summary 元素提供标题或图例。标题是可见的，用户单击标题时，会显示出细节信息。summary 元素应该是 details 元素的第一个子元素。

HTML 5 中代码示例：

```
<details>
<summary>……</summary>
…….
</details>
```

【例 2.24】使用 details 元素制作简单页面（实例文件：ch02\2.24.html）

```
<!DOCTYPE HTML>
<html>
<body>
<details>
  <summary>苹果冰激凌</summary>
  <img src="images/冰激凌.jpg" alt="苹果冰激凌"/>
  <div>
     <h3>材料：苹果 500g，白糖 150g，新鲜牛奶两瓶。</h3>
     <p>制作方法：将苹果洗净，去皮挖核，切成薄片，搅成浆状，放入白糖及 1000 克开水，加入煮沸的牛奶，搅拌均匀，倒入盛器内冷却后置于冰箱冻结即成。
    </p>
  </div>
```

```
</details>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 2-24 所示。

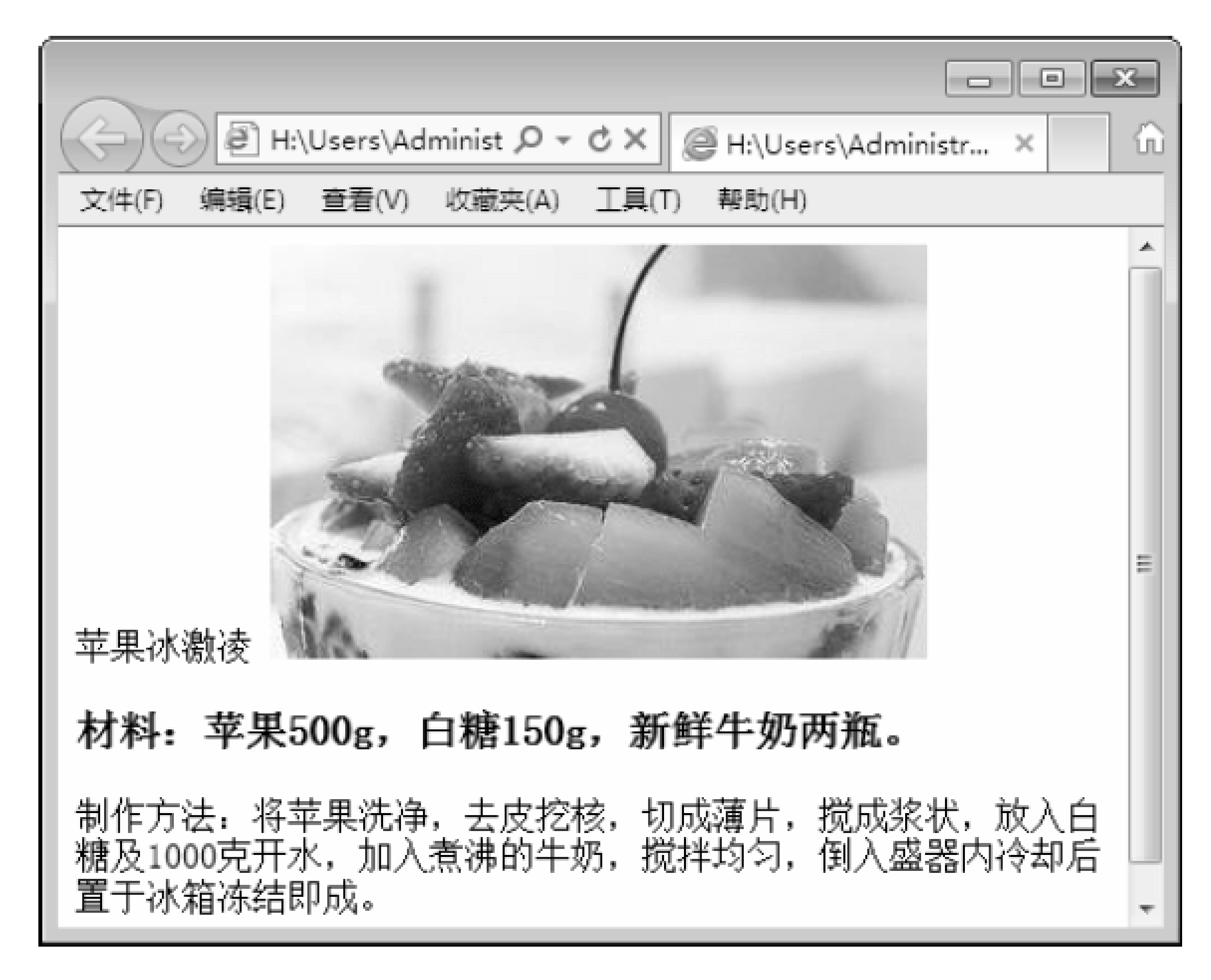


图 2-24　程序运行结果



提　示

默认情况下，浏览器支持 details 元素，除了 summary 标签外的内容将会被隐藏。

2.3.7　datalist 元素

datalist 用来辅助文本框的输入功能，它本身是隐藏的，与表单文本框中的 list 属性绑定，即可将 list 属性值设置为 datalist 的 ID 号，类似 suggest 组件。目前只支持 opera 浏览器。

HTML 5 中代码示例：

```
<datalist></datalist>
```

【例 2.25】使用 datalist 元素制作下列列表框（实例文件：ch02\2.25.html）

```
<!DOCTYPE HTML>
<html>
<head>
    <title>datalist 测试</title>
</head>
<body>
<form action="#">
    <fieldset>
        <legend>请输入职业</legend>
        <input type="text" list="worklist">
```

```
        <datalist id="worklist">
            <option value="程序开发员"></option>
            <option value="系统架构师"></option>
            <option value="数据维护员"></option>
        </datalist>
    </fieldset>
</form>
</body>
</html>
```

在 Opera 中预览效果如图 2-25 所示。



图 2-25　程序运行结果

2.4　新增全局属性

在 HTML 5 中新增了许多全局属性，下面来详细介绍常用的新增属性。

2.4.1　contenteditable 属性

Contenteditable 属性是 HTML 5 中新增的标准属性，其主要功能是指定是否允许用户编辑内容。该属性有两个值：true 和 false。

Contenteditable 属性为 true 时表示可以编辑，为 false 时表示不可编辑。如果没有指定值则会采用隐藏的 inherit（继承）状态，即如果元素的父元素是可编辑的，则该元素就是可编辑的。

【例 2.26】使用 contenteditable 属性的实例（实例文件：ch02\2.26.html）

```
<!DOCTYPE html>
<head>
<title>conentEditalbe 属性示例</title>
</head>
<body>
```

```
<h3>对以下内容进行编辑内容</h3>
<ol contentEditable="true">
<li>列表一</li>
<li>列表二</li>
<li>列表三</li>
</ol>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，打开后可以在网页中输入相关内容，效果如图 2-26 所示。



图 2-26　程序运行结果



对内容进行编辑后，如果关闭网页，编辑的内容将不会被保存。如果想要保存其内容，只能把该元素的 innerHTML 发送到服务器端进行保存。

2.4.2　spellcheck 属性

spellcheck 属性是 HTML 5 中的新属性，规定是否对元素内容进行拼写检查。可对以下文本进行拼写检查：类型为 text 的 input 元素中的值（非密码）、textarea 元素中的值、可编辑元素中的值。

【例 2.27】使用 spellcheck 属性的实例（实例文件：ch02\2.27.html）

```
<!DOCTYPE html>
<html>
<head>
<title>hello，Word</title>
</head>
<body>
<p contenteditable="true" spellcheck="true">使用 spellcheck 属性，使段落内容可被编辑。</p>
```

```
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，打开后可以在网页中输入相关内容，效果如图 2-27 所示。

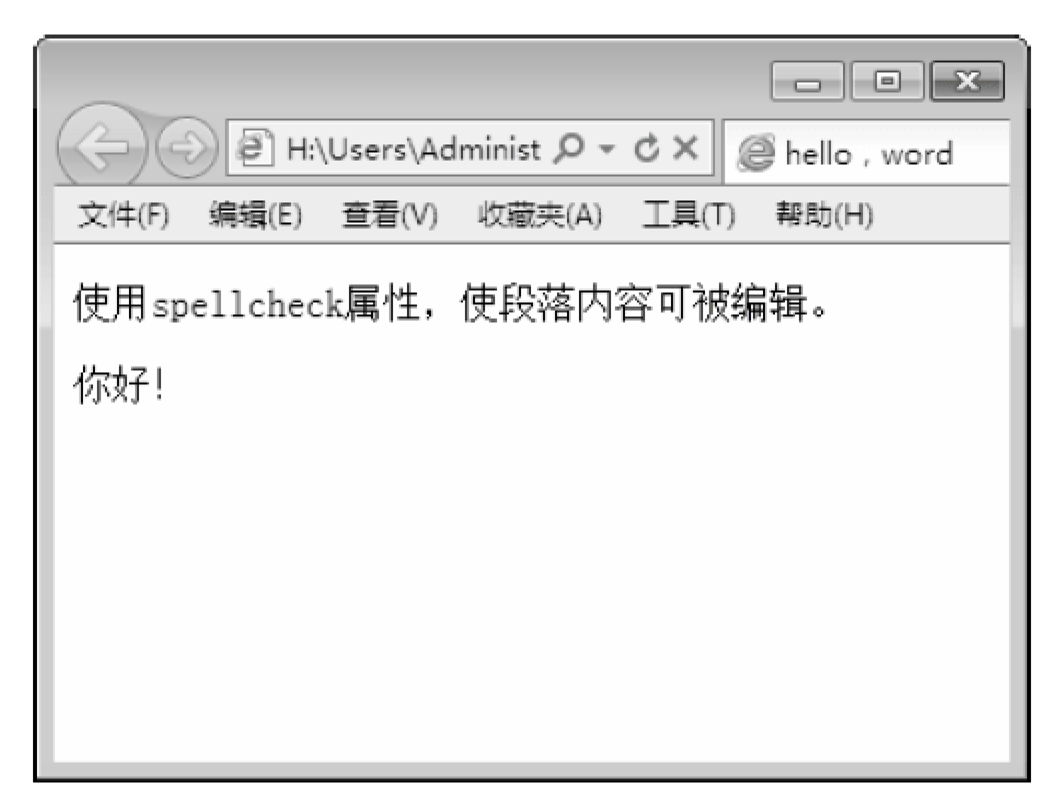


图 2-27　程序运行结果

2.4.3　tabIndex 属性

tabIndex 属性可设置或返回按钮的 Tab 键控制次序。打开页面，连续按下 Tab 键，会在按钮之间切换，tabIndex 属性则可以记录显示切换的顺序。

【例 2.28】使用 tabIndex 属性的实例（实例文件：ch02\2.28.html）

```
<html>
<head>
<script type="text/JavaScript">
function showTabIndex()
{
var bt1=document.getElementById('bt1').tabIndex;
var bt2=document.getElementById('bt2').tabIndex;
var bt3=document.getElementById('bt3').tabIndex;
document.write("Tab 切换按钮 1 的顺序: " + bt1);
document.write("<br />");
document.write("Tab 切换按钮 2 的顺序: " + bt2);
document.write("<br />");
document.write("Tab 切换按钮 3 的顺序: " + bt3);
}</script>
</head>
<body>
<button id="bt1" tabIndex="1">按钮 1</button><br />
<button id="bt2" tabIndex="2">按钮 2</button><br />
<button id="bt3" tabIndex="3">按钮 3</button><br />
```

```
<br />
<input type="button" onclick="showTabIndex()" value="显示切换顺序" />
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，打开后多次按下 Tab 键，使控制中心在几个按钮对象间切换，如图 2-28 所示。



图 2-28　程序运行结果

单击“显示切换顺序”按钮，显示出依次切换的顺序，如图 2-29 所示。

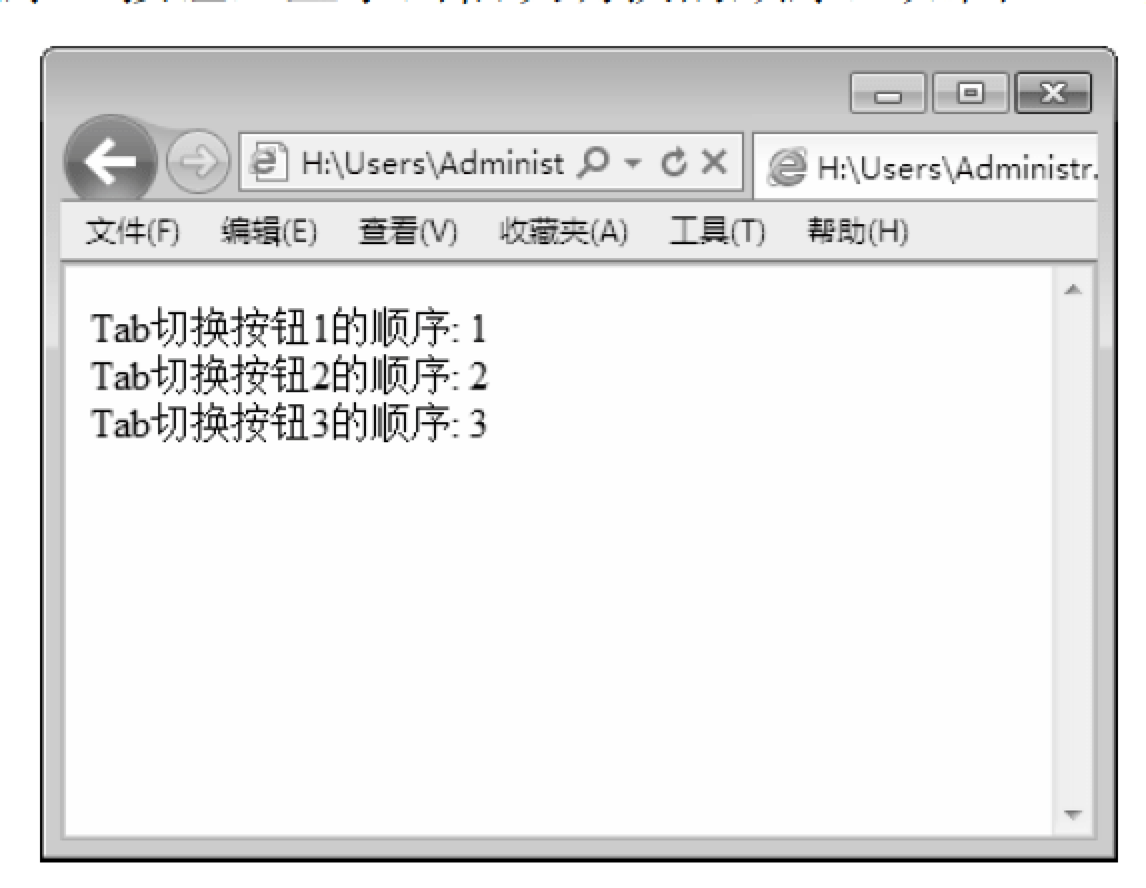


图 2-29　程序运行结果

2.5　新增的其他属性

新增属性主要分为三大类：表单相关的属性、链接相关属性和其他新增属性。

2.5.1　表单相关的属性

新增的表单属性有很多，下面来分别进行介绍。

1. autocomplete

autocomplete 属性规定 form 或 input 域应该拥有自动完成功能。autocomplete 适用于<form>标签，以及以下类型的<input>标签：text、search、url、telephone、email、password、datepickers、range 以及 color。

【例 2.29】使用 autocomplete 属性的实例（实例文件：ch02\2.29.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo form.asp" method="get" autocomplete="on">
    姓名:<input type="text" name="姓名" /><br />
    性别:<input type="text" sex="性别" /><br />
    邮箱:<input type="email" name="email" autocomplete="off" /><br />
  <input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-30 所示。

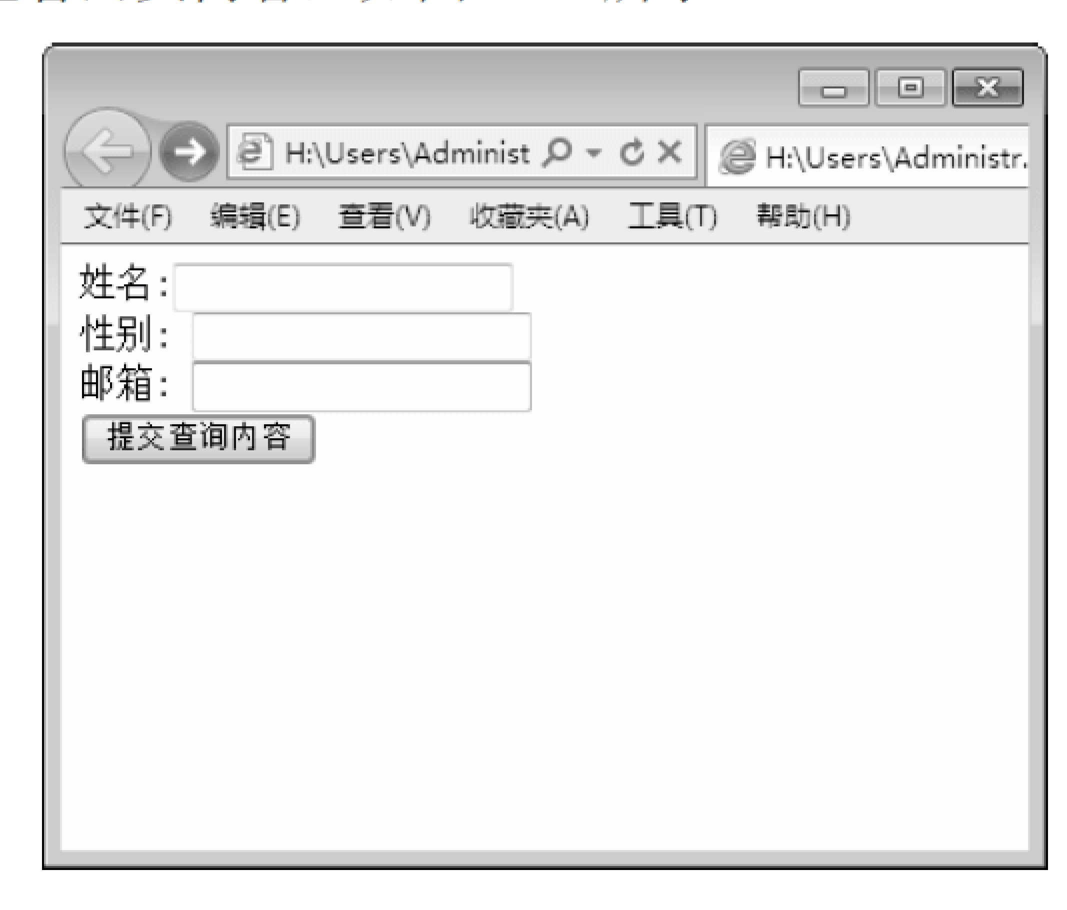


图 2-30　程序运行结果

2. autofocus

autofocus 属性规定在页面加载时，域自动地获得焦点。autofocus 属性适用于所有<input>标签的类型。

【例 2.30】使用 autofocus 属性的实例（实例文件：ch02\2.30.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
```

```
        用户名:<input type="text" name="user_name" autofocus="autofocus" />
        <input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-31 所示。

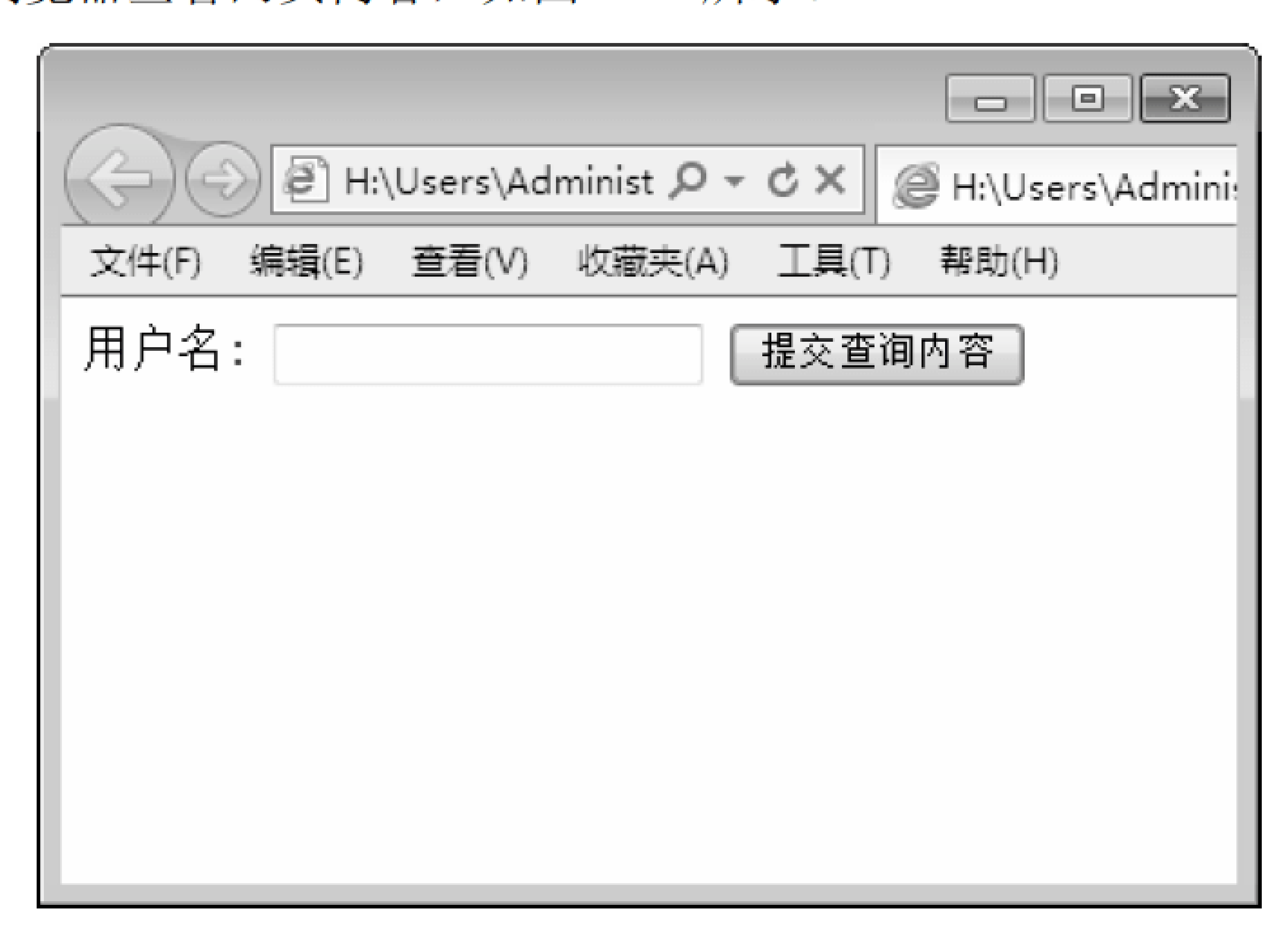


图 2-31　程序运行结果

3. form

form 属性规定输入域所属的一个或多个表单。form 属性适用于所有<input>标签的类型，必须引用所属表单的 id。

【例 2.31】使用 form 属性的实例（实例文件：ch02\2.31.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo form.asp" method="get" id="user form">
        姓名:<input type="text" name="姓名" />
        <input type="submit" />
</form>
        性别:<input type="text" sex="性别" form="user_form" />
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-32 所示。

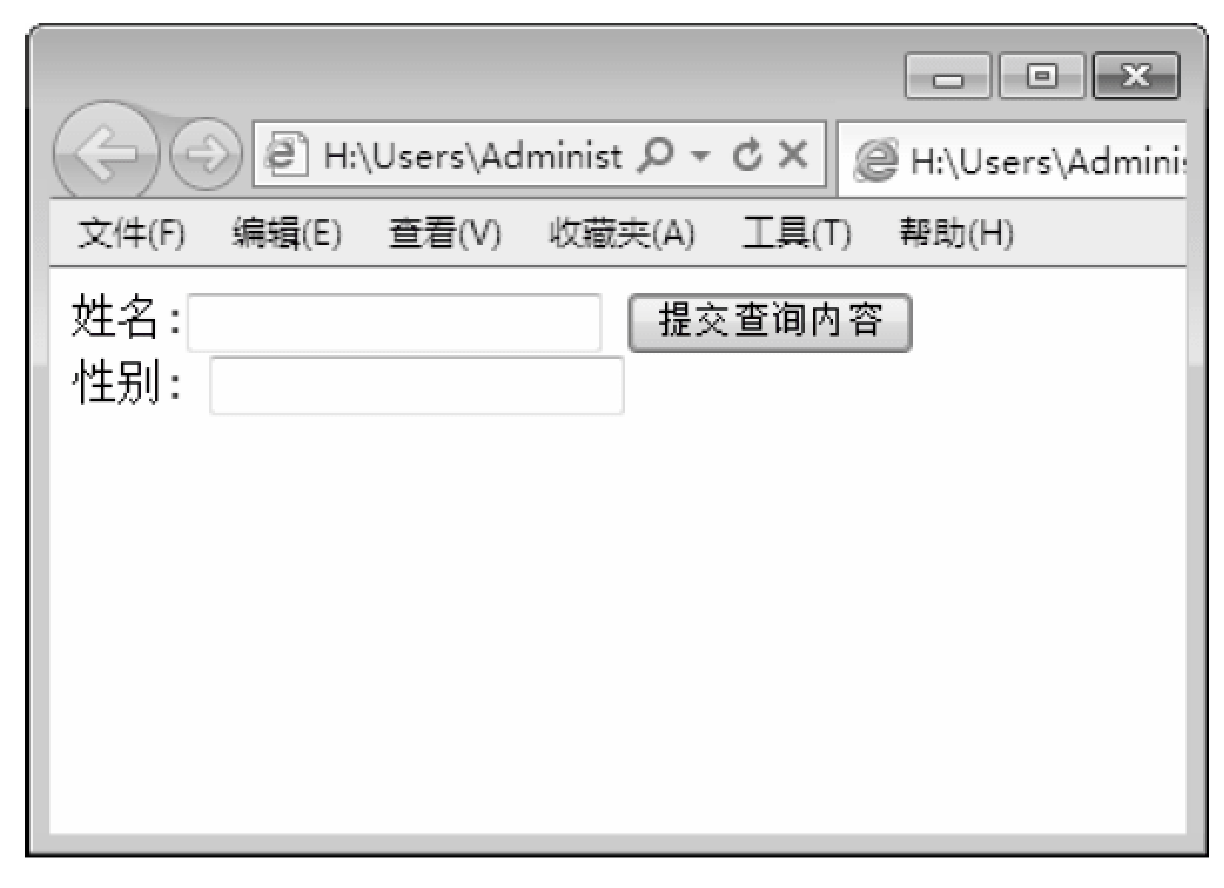


图 2-32　程序运行结果

4. form overrides attributes

表单重写属性（form override attributes）允许重写 form 元素的某些属性设定。

表单重写属性如下。

- formaction: 重写表单的 action 属性
- formenctype: 重写表单的 enctype 属性
- formmethod: 重写表单的 method 属性
- formnovalidate: 重写表单的 novalidate 属性
- formtarget: 重写表单的 target 属性

表单重写属性适用于以下类型的<input>标签：submit 和 image。

【例 2.32】使用 form overrides 属性的实例（实例文件：ch02\2.32.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get" id="user_form">
     邮箱:<input type="email" name="userid" /><br />
   <input type="submit" value="提交" /><br />
   <input type="submit" formaction="demo_admin.asp" value="以管理员身份提交" /><br />
   <input type="submit" formnovalidate="true" value="提交未经验证" /><br />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-33 所示。

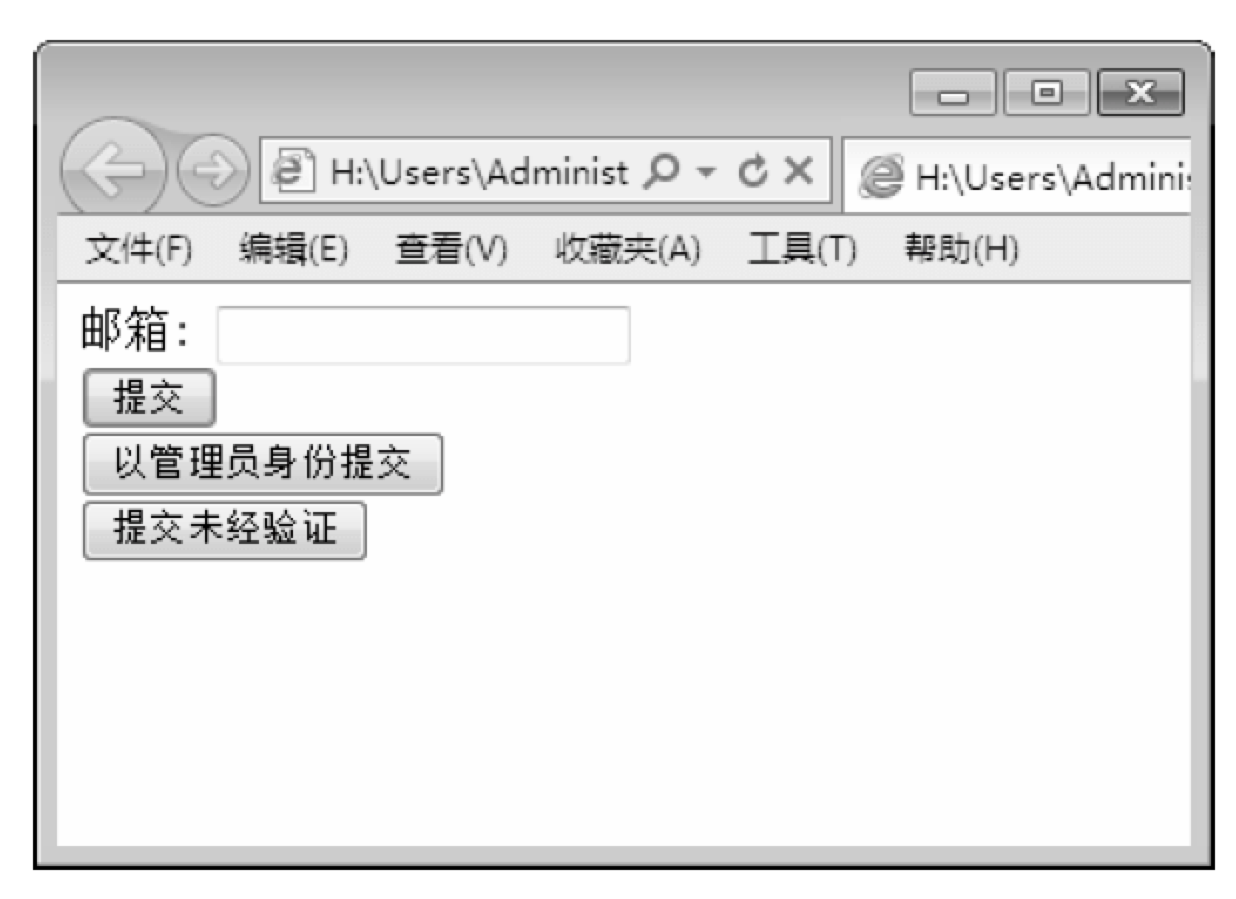


图 2-33 程序运行结果

5. height 和 width

height 和 width 属性用于规定 image 类型的 input 标签的图像高度和宽度。

【例 2.33】使用 height 和 width 属性的实例（实例文件：ch02\2.33.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    用户名:<input type="text" name="user_name" /><br />
   <input type="image" src="/images/按钮.jpg" width="99" height="99" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-34 所示。



图 2-34 程序运行结果

6. list

list 属性规定输入域的 datalist。datalist 是输入域的选项列表。list 属性适用于以下类型的<input>标签：text、search、url、telephone、email、date pickers、number、range 以及 color。

【例 2.34】使用 list 属性的实例（实例文件：ch02\2.34.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    主页:<input type="url" list="url_list" name="link" />
  <datalist id="url_list">
<option label="baisu" value="http://www.baidu.com" />
<option label="qq" value="http://www.qq.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
  </datalist>
<input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-35 所示。

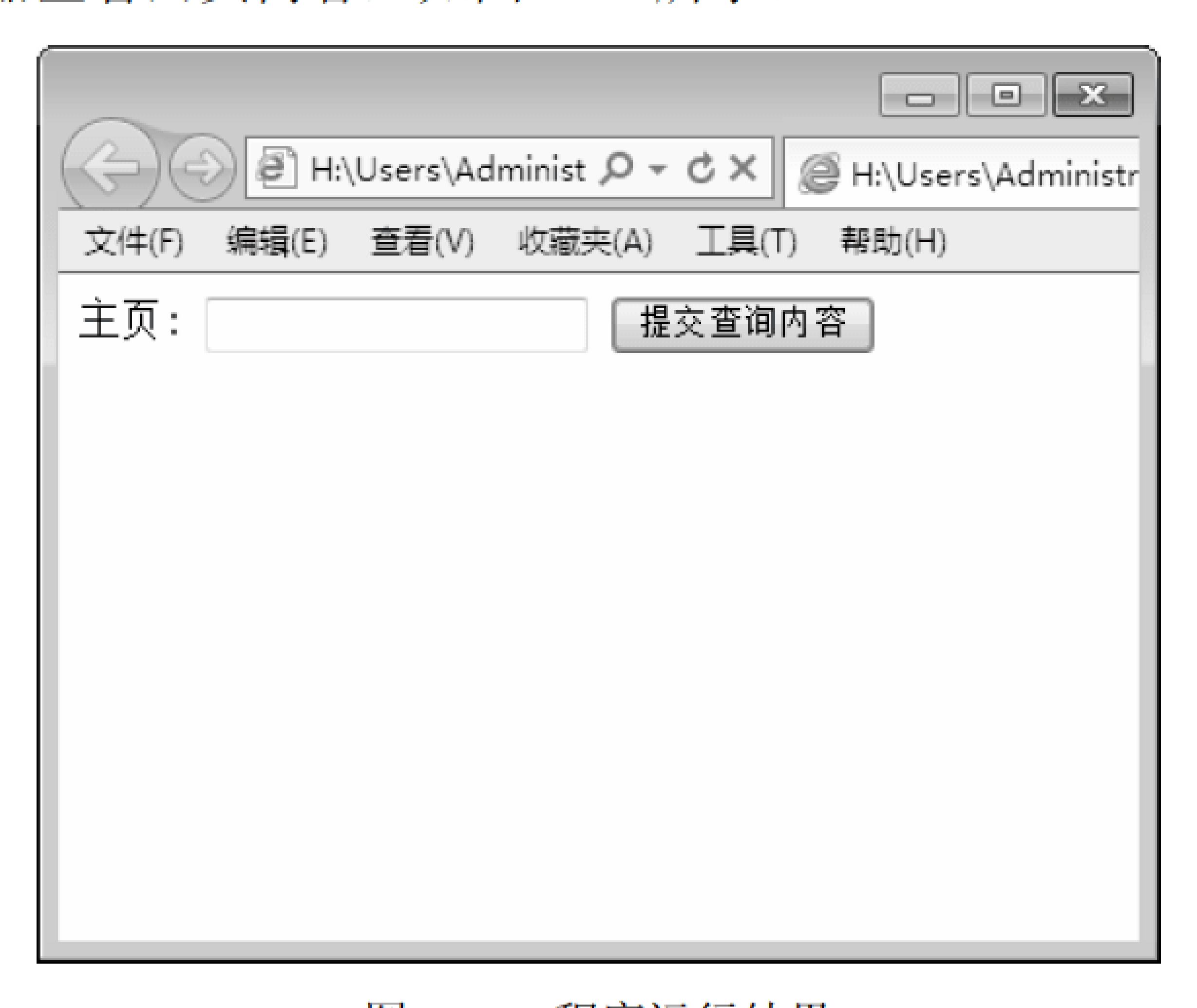


图 2-35　程序运行结果

7. max、min 和 step

max、min 和 step 属性用于为包含数字或日期的 input 类型规定限定（约束）。max 属性规定输入域所允许的最大值；min 属性规定输入域所允许的最小值；step 属性为输入域规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）。

max、min 和 step 属性适用于以下类型的<input>标签：date pickers、number 以及 range。

【例 2.35】使用 min、max 和 step 属性的实例（实例文件：ch02\2.35.html）。

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    成绩:<input type="number" name="points" min="0" max="10" step="3"/>
<input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-36 所示。

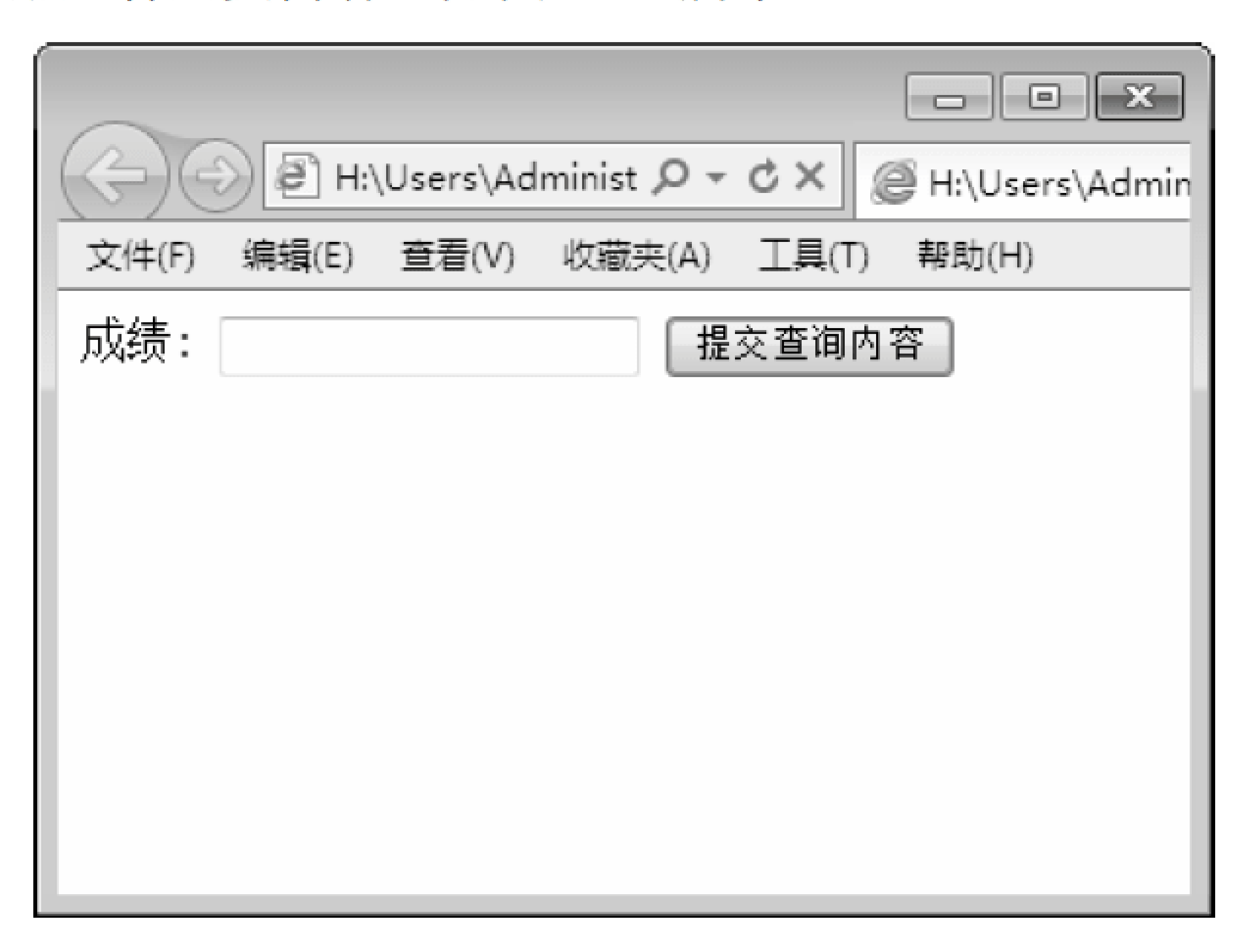


图 2-36　程序运行结果

8. multiple

multiple 属性规定输入域中可选择多个值。multiple 属性适用于以下类型的<input>标签：email 和 file。

【例 2.36】使用 multiple 属性的实例（实例文件：ch02\2.36.html）。

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    选择图片:<input type="file" name="img" multiple="multiple" />
<input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-37 所示。



图 2-37　程序运行结果



单击“浏览”按钮，可以打开“选择要加载的文件”对话框，在其中选择要添加的图片信息。

9. pattern（regexp）

pattern 属性规定用于验证 input 域的模式（pattern）。pattern 属性适用于以下类型的<input>标签：text、search、url、telephone、email 以及 password。

【例 2.37】使用 pattern 属性的实例（实例文件：ch02\2.37.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    电话区号:<input type="text" name="country_code" pattern="[A-z]{3}"
    title="Three letter country code" />
   <input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-38 所示。

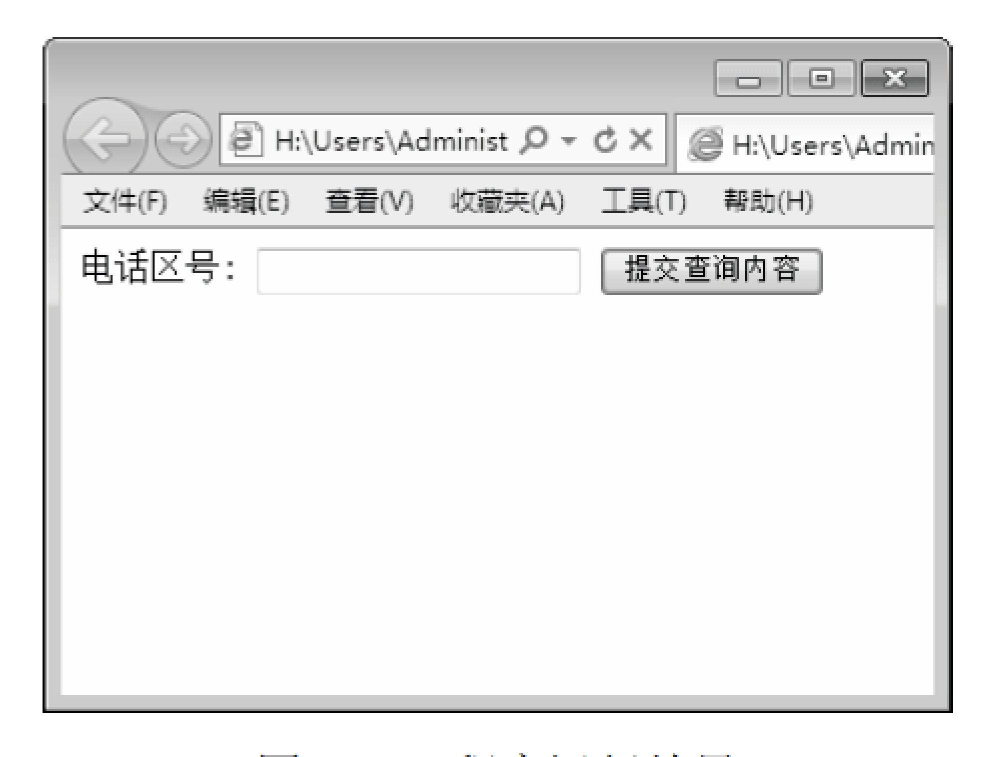


图 2-38　程序运行结果

10. placeholder

placeholder 属性提供一种提示（hint），描述输入域所期待的值。placeholder 属性适用于以下类型的<input>标签：text、search、url、telephone、email 以及 password。

【例 2.38】使用 placeholder 属性的实例（实例文件：ch02\2.38.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    <input type="search" name="user_search" placeholder="baidu" />
    <input type="submit" />
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-39 所示。



图 2-39　程序运行结果

11. required

required 属性规定必须在提交之前填写输入域（不能为空）。required 属性适用于以下类型的<input>标签：text、search、url、telephone、email、password、date pickers、number、checkbox、radio 以及 file。

【例 2.39】使用 required 属性的实例（实例文件：ch02\2.39.html）

```
<!DOCTYPE HTML>
<html>
<body>
<form action="demo_form.asp" method="get">
    姓名:<input type="text" name="usr_name" required="required" />
    <input type="submit" />
```

```
</form>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-40 所示。

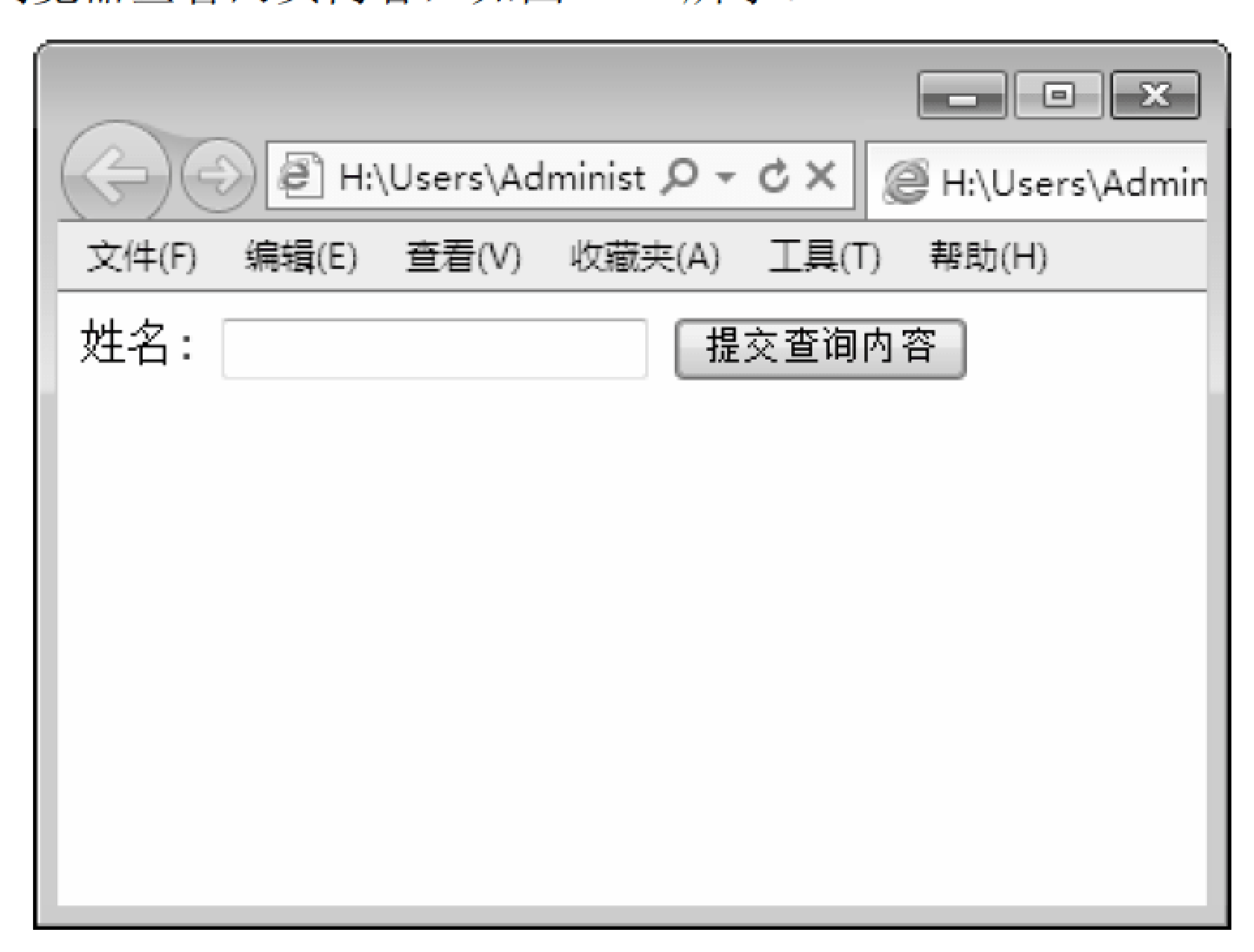


图 2-40　程序运行结果

2.5.2　链接相关属性

下面介绍新增的与链接相关的属性。

1. media 属性

media 属性规定目标 URL 是为哪种类型的媒介/设备进行优化的，只能在 href 属性存在时使用。

【例 2.40】使用 media 属性的实例（实例文件：ch02\2.40.html）

```
<!DOCTYPE HTML>
<html>
<body>
  <a href="www.baidu.com" media="print and (resolution:300dpi)">
      链接查询.
  </a>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-40 所示。

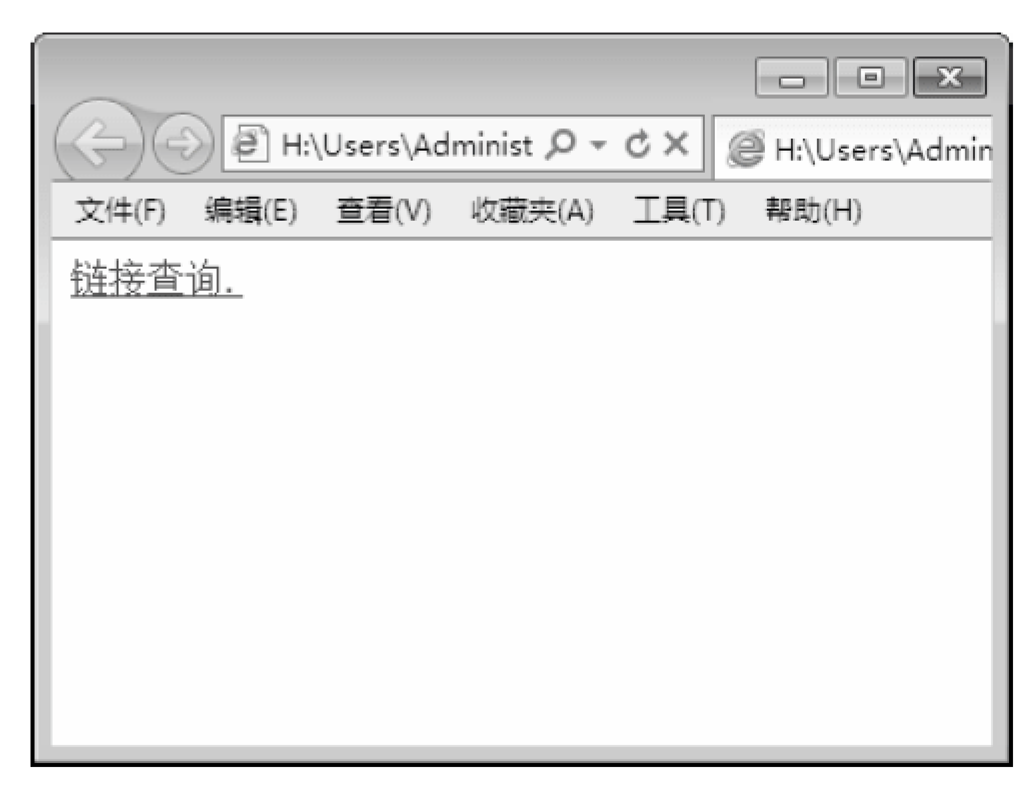


图 2-41 程序运行结果

2. type 属性

在 HTML 5 中，为 area 元素增加了 type 属性，规定目标 URL 的 MIME 类型。仅在 href 属性存在时使用。

语法结构如下：

```
<input type="value">
```

3. sizes

为 link 元素增加了新属性 sizes。该属性可以与 icon 元素结合使用（通过 rel 属性），该属性指定关联图标（icon 元素）的大小。

4. target

为 base 元素增加了 target 属性，主要目的是保持与 a 元素的一致性。

【例 2.41】使用 sizes 与 target 属性的实例（实例文件：ch02\2.41.html）

```
<!DOCTYPE html>
<html>
<head>
    <link rel="icon" href="demo_icon.ico" type="image/gif" sizes="16x16" />
</head>
<body>
    <h2>Hello world!</h2>
    <p>打开<a href="2.40.html" target="_blank">新链接</a>窗口。</p>
</body>
</html>
```

使用 IE 9.0 浏览器查看网页内容，如图 2-42 所示。



图 2-42　程序运行结果

2.5.3　其他属性

除了以上介绍的与表单和链接相关的属性外，HTML 5 还增加了其他属性，如下表所示。

属性	隶属于	意义
reversed	ol 元素	指定列表倒序显示
charset	meta 元素	为文档字符编码的指定提供一种比较良好的方式
type	menu 元素	让菜单可以以上下文菜单、工具条与列表菜单的三种形式出现
label	menu 元素	为菜单定义可见的标注
scoped	style 元素	用来规定样式的作用范围，譬如只对页面上某个树起作用
async	script 元素	定义脚本是否异步执行
manifest	html 元素	开发离线 Web 应用程序时与 API 结合使用，定义一个 URL，在这个 URL 上描述文档的缓存信息
sandbox、srcdoc 与 seamless	iframe 元素	用来提高页面安全性，防止不信任的 Web 页面执行某些操作

2.6　废除的属性

在 HTML 5 中废除了很多不需要再使用的属性，这些属性将采用其他属性或其他方案进行替代，具体内容如下表所示。

废除的属性	使用该属性的元素	在 HTML 5 中代替的方案
rev	link,a	rel
charset	link,a	在被链接的资源中使用 HTTP content-type 头元素
shape，coords	a	使用 area 元素代替 a 元素
longdesc	img，iframe	使用 a 元素链接较长描述
target	link	多余属性，被省略

（续表）

废除的属性	使用该属性的元素	在 HTML 5 中代替的方案
Nohref	area	多余属性，被省略
profile	head	多余属性，被省略
version	html	多余属性，被省略
name	img	id
scheme	meta	只为某个表单域使用 scheme
archive,classid,codebase,codetype,declare,standby	object	使用 data 与 type 属性类调用插件。需要使用这些属性来设置参数时，使用 param 属性
valuetype,type	param	使用 name 与 value 属性，不声明值的 MIME 类型
axis,abbr	td,th	使用以明确简洁的文字开头，后跟详述文字的形势。 可以对更详细的内容使用 title 属性，来使单元格的内容变得简短
scope	td	在被链接的资源中使用 HTTP Content-type 头元素
align	caption,input,legend,div,h1,h2,h3,h4,h5,h6,p	使用 CSS 样式表进行替代
alink,link,text,vlink,background,bgcolor	body	使用 CSS 样式表进行替代
align,bgcolor,border,cellpadding,cellspacing,frame,rules,width	table	使用 CSS 样式表进行替代
align,char,charoff,height,nowrap,valign	tbody,thead,tfoot	使用 CSS 样式表进行替代
align,bgcolor,char,charoff,height,nowrap,valign,width	td,th	使用 CSS 样式表进行替代
align,bgcolor,char,charoff,valign	tr	使用 CSS 样式表进行替代
align,char,charoff,valign,width	col,colgroup	使用 CSS 样式表进行替代
align,border,hspace,vspace	object	使用 CSS 样式表进行替代
clear	br	使用 CSS 样式表进行替代
compact,type	ol,ul,li	使用 CSS 样式表进行替代
compact	dl	使用 CSS 样式表进行替代

（续表）

废除的属性	使用该属性的元素	在 HTML 5 中代替的方案
Compact	menu	使用 CSS 样式表进行替代
width	pre	使用 CSS 样式表进行替代
align,hspace,vspace	img	使用 CSS 样式表进行替代
align,noshade,size,width	hr	使用 CSS 样式表进行替代
align,frameborder,scrollingmarginheight,marginwidth	iframe	使用 CSS 样式表进行替代
autosubmit	menu	

2.7 问题解答

1．HTML 5 中的单标记和双标记书写方法是什么？

HTML 5 中的标记分为单标记和双标记。所谓单标记是指没有结束标记的标签，双标记是指既有开始标记又有结束标记的标签。

对于单标记不允许写结束标记的元素，只允许使用“<元素/>”的形式进行书写。例如“
…</br>”的书写方式是错误的，正确的书写方式为
。当然，在 HTML 5 之前的版本中
这种书写方法可以被沿用。HTML 5 中不允许写结束标记的元素有：area、base、br 、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr。

对于部分双标记可以省略结束标记。HTML 5 中允许省略结束标记的元素有 li、dt、dd、p、rt、rp、optgroup、option、colgroup、thead、tbody、tfoot、tr、td、th。

HTML 5 中有些元素还可以完全被省略。即使这些标记被省略了，该元素还是以隐式的方式存在的。HTML 5 中允许省略全部标记的元素有：html、head、body、colgroup、tbody。

2. 新增属性 Target 在 HTML 4.01 与 HTML 5 之间的差异有哪些？

在 HTML 5 中，不再允许把框架名称设定为目标，因为不再支持 frame 和 frameset。Self、parent 以及 top 这三个值大多数时候与 iframe 一起使用。

第 3 章　认识网页与网站

在高速发展的今天，网站已经成为展示自我，宣传自我的工具。公司或个人可以拥有自己的网站，可以向别人介绍自己，展示自己，这些都离不开 HTML 网页。

3.1　网站的基本概念

当进入一个网站时，首先呈现出来的就是网页，网页是承载各种网站信息的平台。网页上可以有图片、文字、音频和视频等构成网站的基本元素，

3.1.1　什么是网页

网页（Web page）是一个文件，它存放在计算机中，而这部计算机必须是与互联网相连的。网页是由网址（URL，例如 www.sohu.com）来识别与存取的，当在浏览器中输入网址后，网页文件会被传送到正在浏览网页的计算机中，然后再通过浏览器对网页内容进行解释，再展示给网页浏览者，网页通常是 HTML 格式（文件扩展名为.html 或.htm）。

在一个网站中，网页按照类型不同，可以分为两种，主页和普通网页。主页（Home Page）是进入网站的开始画面，如搜狐网站的首页，如图 3-1 所示。



图 3-1　搜狐网站主页

3.1.2　什么是网站

网站（Website）是指根据一定的规则，使用 HTML 等工具制作的用于展示特定内容的相关网页的集合，例如常见的网站有搜狐、新浪等。

简单地说，网站就像布告栏一样，人们可以通过网站发布资讯信息，或者利用网站提供的

相关的网络服务。衡量网站的性能通常从网站空间大小、网站位置、网站连接速度（俗称“网速”）、网站软件配置、网站提供的服务等几方面考虑，最直接的衡量标准是网站的真实流量。

3.2 网页基本构成元素

在早期，网站只能保存单纯文本。经过不断地发展，图像、声音、动画、视频和 3D 技术等已经在因特网上广泛应用，并且通过动态网页技术，用户可以与其他用户或者网站管理者进行交流。

常见的网页构成元素有文本、图像、超链接、表格、表单、导航栏、动画、框架等，如图 3-2 所示。



图 3-2 网页常见构成元素

- 文本：网页中的信息主要以文本为主。在网页中可以通过字体、大小、颜色、底纹、边框等来设计文本的属性。
- 图像：有了图像，才能看到丰富多彩的网页。网页上图片多为 JPG 或 GIF 格式。通常图片会被运用在 Logo、Banner 和背景图上。Logo 是代表企业形象或栏目内容的标志性图片，一般在网页的左上角。Banner 是用于宣传网站内某个栏目或活动的广告，一般要求制作成动画形式，达到宣传的效果。Banner 一般位于网页的顶部和底部。在网页页面中比较常用的图片还包括背景图，但要慎用。
- 超链接：超链接是网站的灵魂，从一个网页指向另一个目的端的链接。目的端通常是一个网页，但也可以是一幅图片、一个电子邮件地址、一个文件、一个程序或者是本网页中的其他位置，超链接本身可以是文字或者图片。
- 表格：是网页中展现数据的主要方式，能够以表的形式显示数据信息。表格也可以用作网页排版，在很长一段时间内，使用表格排版是网站的首选方式。
- 表单：表单是用来收集站点访问者信息的集合。站点访问者填写表单的方式是输入文本、单击单选按钮与复选框，以及从下拉菜单中选择选项。在填好表单后，站点访问者便送出所输入的数据，该数据就会根据网站设计者所设置的表单处理程序，以各种

不同的方式进行处理。

- 导航栏：导航栏是一组超链接，一般用于网站各部分内容间相互链接的指引。导航栏可以是按钮或者文本超链接。
- 动画：动画是网页上最活跃的元素，包括 GIF 动画和 Flash 动画。



网页中除了这些最基本元素外，还包括横幅广告、字幕、悬停按钮、计数器、音频和视频等。

3.3 网页设计

网页设计，是网站建设行业内的专业术语，作为客户，若能了解更多的网站设计基础知识，在进行网站建设过程中就不会显得被动、无助，而且能更好地与项目负责人进行沟通交流，以满足自己的设计需求。

3.3.1 网页设计概述

网页设计是网页创作过程，是建立在 Internet 上的新型设计。它具有很强的视觉效果、互动性、互操作性等其他媒体所不具有的特点。

一个成功的网页设计，首先在观念上要确立动态的思维方式，其次，要有效地将图形引入网页中，增加人们浏览网页的兴趣，在崇尚鲜明个性风格的今天，网页设计应增加个性化因素。例如搜狐网页中的视频网站地图，如图 3-3 所示。



图 3-3 视频地图显示

网页设计融合了 IT 技术与美术设计两个方面。因此，对从业人员来说，仅掌握 IT 技术是不够的，还需要有一定的美术功底与审美能力。而且，网页设计傻瓜软件正在日益普及，技术早已不是这个领域的唯一门槛。在这种情况下，美术功底和审美能力是决定成为什么级别的网页设计师的关键因素。

3.3.2　网页设计特点

网页设计有下面几个特点。

1．交互性与持续性

网页被创建出来后，上传到网站，读者就可以浏览和欣赏了。通常为了吸引读者注意，需要定时的更新网页内容，例如更新新闻、股票信息等。在更新过程中，其版面设计可能会由于内容量的多少，版面发生变化，这时就需要网页设计者保持一个连续的设计风格。

有的网页为了及时获取最新舆论信息，通常都存在 HTML 留言表单。浏览者登录网页后，发布自己的言论信息。面对每天都在增长的评论内容，网页设计者也需要保持一个版面的设计风格。例如下面页面，对某一主题，浏览者在不同时间进行评论，内容不断增多，但其版面效果不会发生变化，如图 3-4 所示。



图 3-4　发表评论

有的网站在使用了一段时间，其风格不再吸引人，这时就面临着整个网站的改版问题，相对于前面网站不能变化太大，也需要保持视觉形象上的持续性。

与报纸等传统传媒布局设计相比较，可以发现，报纸排版完成后只需出版，不再具有重新排版布局的可能，而网页不一样，由于需要和浏览者不断交互，需要不断排版和布局，甚至每隔一定时间，还需要版本升级。

2．多维性

HMTL 网页最吸引人之处，就是存在超级链接。正是因为有了超级链接，浏览者才可以在不同页面之间，任意跳转。可以想象，如果 HTML 网页，没有了超级链接，那只能按看图书的方式浏览网页，首先在目录中找到这个网页地址，然后再看，看完了再返回去。超级链接的出现，大大丰富了页面之间的联系。超级链接可以将页面组织结构分为序列结构、层次结构、网状结构、复合结构等。例如百度首页，就是层次结构，单击页面任意一个超级链接，可以进入此类网页查看，如图 3-5 所示。



图 3-5　百度网站首页

网站中，一个合理的页面导航可以帮助浏览者迅速找到目标网页。导航需要提供足够的、不同角度的链接，帮助浏览者在网页各个部分跳转，并告知浏览者所在位置、当前页面和其他页面之间的关系等。通过超级链接，可以链接到不同页面，这就是网页设计的多维性特点。

3．多种媒体综合性

在创建网页时，不仅仅要考虑文本元素，还需要考虑图像、音频和视频等元素，只有这样才能将网页设计的动感十足。随着硬件技术的提高，例如网络带宽增加、芯片处理速度等，将来还需要考虑 3D 元素增加。在数据压缩技术的改进和流技术的推动下，网上出现了实时的音频和视频服务，例如在线音乐、网上电影等，如图 3-6 所示。



图 3-6　迅雷网站首页

多种媒体综合运用是网页设计的特点之一，也是网页设计者必须要考虑的元素之一。

4．版式不可控性

网页版式设计存在许多不可控因素，首先网页设计者创建的网页，在不同的浏览器运行时，可能会存在差异。其次，网页技术在不断发展，如有一个网站，在 Firefox 浏览器中显示，如

图 3-7 所示。可以发现图片周边区域间隔比较大，但在 IE 浏览器中会发现，图片四周没有间隙，如图 3-8 所示。



图 3-7　用 Firefox 浏览器显示



图 3-8　用 IE 浏览器显示

这一切说明，网络应用尚处在发展之中，很难制订出统一的标准，必然导致网页版式设计的不可控性。其具体表现为：

- 网页页面会根据当前浏览器窗口大小，自动格式化输出。
- 网页浏览者可以控制网页在浏览器中的显示方式。
- 用不同种类、版本的浏览器观察同一个网页页面时，效果会有所不同。
- 用户浏览器工作环境不同，显示效果也会有所不同。

把所有这些问题归结为一点，即网页设计者无法控制页面在用户端的最终显示效果，但这也正是网页设计的吸引人之处。

5. 技术与艺术结合的紧密性

设计是主观和客观共同作用的结果，是在自由和不自由之间进行的，设计者不能超越自身已有经验和所处环境提供的客观条件限制，优秀设计者正是在掌握客观规律基础上得到了完全的自由，拥有了想象和创造的自由。

在网页设计的过程中，网络技术主要表现为客观因素，艺术创意主要表现为主观因素，网页设计者应该积极主动地掌握现有的各种网络技术规律，注重技术和艺术紧密结合，这样才能穷尽技术之长，实现艺术想象，满足浏览者对网页信息的高质量需求。

例如，浏览者要欣赏一段音乐或电影，以前必须先将这段音乐或电影下载到本地计算机的磁盘当中，然后使用相应的程序来播放才能实现，又由于音频或视频文件都比较大，需要较长的下载时间。但是，当流（Stream）技术出现以后，网页设计者充分、巧妙地应用此技术，让浏览者在下载过程中就可以欣赏这段音乐或电影，实现了实时的网上视频直播服务和在线欣赏音乐服务，无疑增强了页面传播信息的表现力和感染力，如图 3-9 所示。



图 3-9　技术和艺术相结合

3.3.3　网页设计相关术语

学习网页设计，会涉及到一些专业术语，本节将会对这些术语进行详细介绍。

1．万维网

万维网（World Wide Web）缩写 WWW 或简称 3W，它是无数个网络站点和网页的集合，也是 Internet 提供的最主要的服务。它是由多媒体链接而形成的，当用户单击超级链接时，会通过 URL 直接指向要显示页面，如图 3-10 所示。



图 3-10　万维网的应用

2．Web 服务器和客户端

WWW 基于客户机/服务器模式，与平台无关。客户机和服务器都是独立的计算机，客户机是浏览者使用的机器，服务器是存放网络服务的计算机。如果一台连入网络的计算机，向其他计算机提供各种网络服务（如数据、文件的共享等）时，它就被称作服务器。而那些用于访问服务器资料的计算机则被称作客户机，其结构模式如图 3-11 所示。

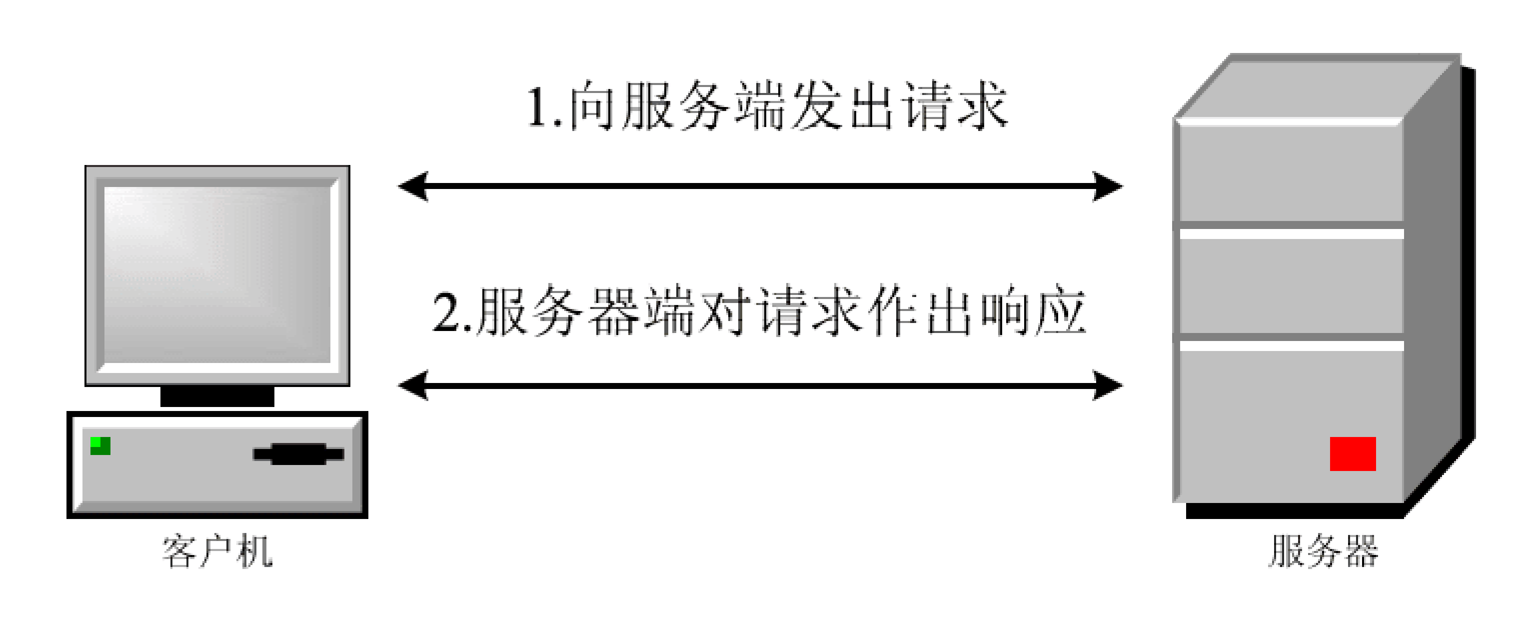


图 3-11　客户机与服务器模式

通常，服务器对于浏览 Web 站点的用户是透明的，Web 服务器传送 HTML 页面使浏览器可以浏览。确切一点，Web 服务器专门处理客户机请求（request），处理完请求之后，会将这些请求发送到客户端。例如返回一个 HTML 页面，一个图像等。为了处理一个请求（request），Web 服务器可以响应（response）一个静态页面或图片，进行页面跳转（redirect）。客户机主要享受网络上提供的各种资源。

3．客户端浏览器

浏览器是一种软件，可以显示网页文件内容，例如显示文本信息、显示图像、播放音频和视频等，大部分的浏览器能够扩展众多插件（plug-ins）。另外，许多浏览器还支持其他的 URL 类型及其相应的协议，例如 FTP、Gopher、HTTPS（HTTP 协议的加密版本）。

HTTP 内容类型和 URL 协议规范允许网页设计者在网页中嵌入图像、动画、视频、声音、流媒体等。浏览器在操作系统中是独立存在，读者可以根据喜好，在操作系统中安装不同的浏览器软件。现在比较流行的浏览器有 IE 和 Firefox 等。

4．统一资源定位符（URL）

URL（Uniform Resource Locator），称为统一资源定位符，它是一个能识别 Internet 地址的信息资源。通过 URL 可以到达任何网络地址寻找所需的资源，例如文件、数据库、图像、新闻组等等。例如 http://www.sohu.com/index.html 就是一个常见的 URL 资源。

3.3.4　网页设计原则

网页的艺术设计，是技术与艺术的结合，内容与形式的统一。它要求设计者必须掌握以下三个主要原则。

1．主题鲜明

网页需要有明确的主题，并能够以独特的样式吸引浏览者的注意。要求在视觉设计上一定要有目标，有明确主题，并能在最短时间内被浏览者接受。视觉设计需要具备目标唯一、简单明了、精确和清晰等特点，最好还要具备一定的视觉冲击效果。

网站主题决定了网页设计风格，只有主题表达清楚，才能添加艺术信息。一个网站视觉上再美，如果没有主题信息，也不会有浏览者访问。

例如某个旅游网站，所展现的都应是与旅游相关的信息，如图 3-12 所示。



图 3-12　主体明确

2．形式与内容统一

俗话说表里如一，用在网页设计中是非常恰当的。任何一个网站都存在内容和形式，内容就是要表达的对象，形式就是内容的表现方式，例如一个新闻，新闻本身是就是内容，具有极高的吸引价值，而形式就是新闻显示的方式和样式，如文本显示，此时需要布局、搭配颜色。

在网页设计中，内容可以说是主题、题材等元素总和，形式就是它的结构、风格和设计语言等。内容主宰形式，形式影响内容。形式必须符合内容，二者才能相得益彰。如果只追求花哨的表现形式以及惊人的冲击效果，或者只是追求内容而缺乏任何的艺术表现，网页设计要么会变得空洞无力，要么会变得单一乏味。还需要文字和图形都表现一个中心，如图 3-13 所示。



图 3-13　内容形式统一

3．强调整体

网页整体效果，即整体风格，是网页设计者必须要追求的。例如一个人如果上身穿西装，下身穿裤头，脚上穿雨靴，就没有整体形象了。网页是一种媒体表现形式，是用来传播信息的。它所承载的信息，必须具备一定价值，有一定的内容、主题和意念，这样才能够被人们所接受，从而满足人们需求。

然而，过分追求整体效果，会牺牲页面个体的灵活性。所以，在强调网页整体设计的同时，还应注意网页个体灵活性，不然网页就会显得呆板、沉闷、枯燥，进而影响访问者的情趣或浏

览欲望。

3.3.5　网页设计的成功要素

网站设计成功的要素有整体布局、有价值的信息、速度、图形、版面设计、文字可读性、网页标题可读性、网站导航、保护个人信息声明和词语准确。

1．整体布局

网页是访问者最先看到的信息，应该做到能够在第一时间吸引住浏览者。好的网页应该干净整洁、条理清楚、引人入胜。不应是杂乱无章的，例如有过多的闪烁、色彩、下拉菜单框、图片等，这样就会分散浏览者的注意力。

2．有价值的信息

无论任何种类的网站，其目的都是给读者一定的价值信息，这样读者才会愿意浏览。这些有价值的信息，可以是新闻、娱乐等。如果是企业网站，需要提供关于产品信息和服务信息，这些信息必须容易理解和查询。

3．速度

如果 30 秒内还打不开网页，一般人就会没有耐心。因此，在设计网页的过程中，至少应该确保主页速度尽可能快，最好不要用大的图片。

4．图形和版面设计

图形和版面设计关系到浏览者对网页的第一印象，图形应集中反映主页所期望传达的主要信息，如图 3-14 所示。



图 3-14　版面设计

5．文字可读性

在设计网页的文字信息时，一定要给文字周围留有足够的空间，才能使浏览者感觉不到压抑；另外，文字的颜色最好不要使用紫色、橙色和红色，因为这些颜色会让人眼花缭乱。

另一种能够提高文字可读性的因素是所选的字体，通用字体最易阅读，特殊字体用于标题效果较好，但是不适合正文。

6．网页标题可读性

网页标题必须容易阅读，需要为所有标题和副标题设置同一字体，并将标题字体加大一号，所有标题和副标题都采用粗体，这样便于识别标题（字体加大加粗）和副标题（粗体，与正文字体大小相同），使浏览者一眼就可以看到要点，以便找出并继续阅读有兴趣的内容。

7．网站导航

由于人们习惯于从左到右、从上到下阅读，所以导航条通常应放置在页面左边，对于较长页面来说，在最底部设置简单导航也很有必要。确定一种满意模式之后，最好将这种模式应用到同一网站的每个页面，这样，浏览者就知道如何寻找信息。

8．词语准确

一个网站如果只有漂亮的外观但是却词语错误连篇、语法混乱，同样是失败的，因此，应注意词语的准确无误。

3.3.6 网页设计风格及色彩搭配

网站设计要注意的两个要点：整体风格和色彩搭配。

1．确定网站整体风格

同一个主题，任何两人都不可能设计出完全一样的网站。风格（style）是抽象的，是指站点的整体形象给浏览者的综合感受。这些都是网站给人们留下的不同感受。例如，如图 3-15 所示的网站首页，特点是风格简洁、大方，主题明确。



图 3-15 整体风格

2．网页色彩搭配

无论是平面设计，还是网页设计，色彩永远是最重要的一环。当距离显示屏较远的时候，看到的不是优美的版式或美丽的图片，而是网页的色彩。例如图 3-16 所示的首页，其颜色以浅色作为主色调，可以衬托文字和图片效果。



图 3-16 色彩搭配

下面介绍一些网页色彩搭配的实战经验。

- 白纸黑字是永远的主题。
- 网页最常用流行色:
  - 蓝色——蓝天白云，沉静整洁的颜色。
  - 绿色——绿白相间，雅致而有生气。
  - 橙色——活泼热烈，标准商业色调。
  - 暗红——宁重、严肃、高贵，需要配黑和灰来压制刺激的红色。
- 颜色忌讳:
  - 忌脏——背景与文字内容对比不强烈，灰暗的背景会令人沮丧。
  - 忌纯——艳丽的纯色对人的刺激太强烈，缺乏内涵。
  - 忌花——要有一种主色贯穿其中，主色并不是面积最大的颜色，而是最重要，最能揭示和反映主题的颜色。
  - 忌粉——颜色浅固然显的干净，但如果对比过弱，就会显得苍白无力。
  - 蓝色忌纯，绿色忌黄，红色忌艳。
- 几种固定搭配:
  - 蓝白橙——蓝为主调。白底，蓝标题栏，橙色按钮或 ICON 做点缀。
  - 绿白兰——绿为主调。白底，绿标题栏，兰色或橙色按钮或用 ICON 做点缀。
  - 橙白红——橙为主调。白底，橙标题栏，暗红或桔红色按钮或用 ICON 做点缀。
  - 暗红黑——暗红主调。黑或灰底，暗红标题栏，文字内容背景为浅灰色。

3.4 网页设计师应具备的素质

当打开网站时，首先映入眼帘的是该网页的界面设计，如内容介绍、按钮摆放、文字组合、色彩应用、网页导航等，这些都是网页设计师的工作。作为优秀的网页设计师，需要具备艺术素质、技能素质和综合素质等。

3.4.1　艺术素质

美是包含或体现社会生活的本质、规律，引起人们的特定情感反映的具体形象（包括社会形象、自然形象和艺术形象）。审美能力是个人所具有的进行审美活动相关的主观条件和心理能力。

作为一名网页设计师，要使自己的网页具有美感。这要求设计师平时多积累，在仔细观察的基础上分析美的来源，并能灵活地将这种美在自己的作品中表现出来。只有这样才能使自己的审美能力达到一定的高度。

网页设计中所蕴含的美，是通过视听来实现的。网页设计中恰当引导访问者的视（文字、图形、动画）听（声音）两种感官，使其获得“美感”，是设计中的关键。例如，如图 3-17 所示的网页首页，通过色彩、图片和创意将美发挥出来。



图 3-17　艺术审美

3.4.2　技能素质

技术问题是网页设计的核心。网页设计者不仅要考虑设计给用户的感觉、用户访问站点的目的，同时还要考虑如何运用技术把自己的思想表达出来。

3.4.3　综合素质

作为合格的网页设计师，必要的文化素质是不可少的。当然文化素质不仅仅包括文学修养，还有音乐和绘画等方面的修养。只有具备这些方面的修养，才能够使自己网页达到一定的水准，才能够使人欣赏到好的网页制作。

网页设计者的素质不是在短时间内就能提高的，要靠时间用心积累而成。因此，要想成为一名真正的设计高手，网页设计者不仅要有一颗真诚为浏览者服务的心，还要广泛涉猎各个领域，特别是要提高自己的艺术修养、文学水平以及鉴别能力。

3.5 网站制作流程

网站的整体规划和设计的好坏是它发展的重要之处，也是它吸引人们浏览的所在之处。动手创建网站，需要遵循一定制作流程。

3.5.1 前期策划

在做网站之前，要有准确的定位。在明确建站目的和网站定位以后，就要开始收集相关意见。

1. 网站定位

如果是公司的网站，在定位时，要与公司决策层人士共同讨论，以便于让上层领导能对网站发展方向有一定的把握，同时最好调动公司其他部门一起参于讨论，能够及时从公司立场提出好的建议，并结合到策划中去。策划时全面考虑，是前期策划中最为关键的一步。

2. 收集相关信息

网站如果是为公司服务的，就要收集各个部门的意见和想法这是必要的，需要将这一步整理成文档，并找出重点。再根据重点以及公司业务侧重点，结合网站定位来确定网站分栏有哪几项。

3.5.2 页面细化及实施

页面设计人员是根据策划书来设计页面的，在设计之前应该让栏目负责人把需要特殊处理的地方和设计人员讲明。在设计页面时，设计人员一定要根据策划书把每个栏目的具体位置和网站的整体风格确定下来，为了让网站有整体感，应该在网页中放置一些贯穿性的元素。

实现设计时，由制作人员负责实现网页，并制作成模版。在这个过程实现的同时，栏目负责人应该开始收集每个栏目的具体内容并进行整理。模版制作完成后，由栏目负责人往每个栏目里面添加具体内容。

在上述过程进行的同时程序员应该正处于开发程序的阶段，要注意选择好的网站开发语言和适合的数据库。

3.5.3 网站上传

将网站上传到网络服务器的整个流程是：首先要在网络服务器上注册域名和申请网络空间，然后配置网站系统，最后才能上传网站。

1. 申请域名

网站建好之后，就要给网站注册标识，即域名。

申请域名的方法很多，用户可以登录域名服务商的网站，根据提示申请域名。域名有免费域名和收费域名两种，用户可以根据实际的需要进行选择。

2. 申请空间

域名注册成功之后，就需要申请网站空间。应根据不同的网站类型选择不同的空间。

网站空间有免费空间和收费空间两种，对于个人网站的用户来说，可以申请免费空间使用。免费空间只需向空间的提供服务器提出的申请，在得到答复后，按照说明上传主页即可，主页的域名和空间都不用操心。使用免费空间美中不足的是：网站的空间有限，提供的服务一般，空间不是非常稳定，域名不能随心所欲。

对于商业网站而言，用户需要考虑空间和安全性等因素，为此可以选择收费网站。

3. 配置网站系统

对于企业单位来说，如果是企业自己的服务器，那么只要把做好的网站（包括 CGI、ASP、JSP 或者 PHP 等程序）发到 WWW 路径下就可以了。而对于个人申请的免费空间网页，就需要将本地计算机上已经做好的网站上传到申请好的网站服务器的免费空间上去。

3.5.4 后期维护

网站制作并完成上传后，接下来是网站维护，下面介绍网站后期维护的主要方法。

1. 做好网站备份

现在互联网技术普及，普通用户利用黑客软件，就能攻击安全性不高的网站，从而造成网站服务器崩溃、数据丢失等，最终网站将付诸东流。因此，对网站的定期备份，不仅是对自己负责，还是对网站忠实用户的尊重。

2. 定期查看网站日志

网站管理人员，应该查看网站日志。通过网站日志，可以了解到搜索引擎何时访问了网站，又访问了哪些页面，经常访问哪些页面，这些信息对进一步优化改善网站非常重要。

此外，维护人员还可以了解网站是否出现了故障，是否已被入侵。最重要的是通过网站日志，可以检测出是否有采集软件在采集网站数据，网站是否被盗链。

3. 网站内容维护

内容是比较重要的。当用户第二次来网站和前一次访问看到的内容一样时或许还回来第三次，但第三次内容还是没有更新，那就可能要失去这个忠实用户了。

4. 网站外链维护

外链是网站维护不可分割的一部分，在互联网中，没有了外链网站就像大海中的孤岛。外链不仅仅是在互联网这个地图上标出了位置，更是为搜索引擎建起了桥梁，当桥梁足够宽阔时，搜索引擎会带来源源不断的客流。

3.6　综合实例——搜集网页素材

目前图片是组成网页的主要元素之一，在浏览网页时，如果遇到比较漂亮的图片，可以将其下载并保存起来，以便以后欣赏和使用。保存网页中图片的具体操作步骤如下：

01 打开一个存在图片的网页，如图 3-18 所示。



图 3-18　美图网页

02 在图片的任意位置单击鼠标右键，从弹出的快捷菜单中选择“图片另存为”菜单项，如图 3-19 所示。



图 3-19　“图片另存为”选项

03 即可打开“保存图片”对话框，在“文件名”文本框中输入要保存图片的名称，单击“保存类型”右侧的下拉按钮，在弹出的下拉列表中选择“JPEG（*.jpg）”选项，如图 3-20 所示。



图 3-20 “保存图片”对话框

04 单击“保存”按钮，即可开始下载并保存图片。打开图片存放的目录，即可在该文件夹下看到下载的图片缩略图，如图 3-21 所示。



图 3-21 保存的图片

3.7 问题解答

1．网页设计中，图像如何使用？

图像内容应有一定的实际作用，切忌虚饰浮夸。图像可以弥补文字的不足，但并不能够完全取代文字。很多用户把浏览软件设定为略去图像，以求节省时间，因此，制作主页时，必须注意将图像所链接的重要信息或链接其他页面的指示用文字重复表达几次，同时要注意避免使

用过大的图像，如果不得不放置大的图像在网站上，应该把图像的缩小版本的预览效果显示出来，这样用户就不必浪费金钱和时间去下载他们根本不想看的大图像了。

2．如何使用动画？

大家都喜欢用 GIF 动画来装饰网页，虽然 GIF 动画的确很吸引人，但在选择时，要判断是否必须用 GIF 动画，如果答案是否定的，那么就选择静态的图片，因为它的容量要小得多。同样尺寸的 Logo，GIF 动画的容量有 5K，而静止 Logo 的只有 3K。虽然只有 2K 之差，但多起来，就会影响下载的速度，所以，如果有些不是必须的，就选择最小的。

3．网页设计对 HTML 代码要求大么？

为了成功地设计网站，必须理解 HTML 是如何工作的。建议新手应从本书中去寻找答案，用记事本制作网页。因为用 HTML 设计网站，可以控制设计的整个过程。

第 4 章　HTML 5 中的文档结构

文档结构，主要是指文章的内部结构，在网页中则表现为整个页面的内部结构。在 HTML 5 之前，并没有对网页文档的结构进行明确的规范，因而打开网页源代码时，可能无法分清哪些是头部哪些是尾部，而在 HTML 5 中则对此做了明确的规范。

4.1　Web 标准

在学习 HTML 5 网页文档结构之前，首先需要了解 Web 的标准，该标准主要是为了解决各种浏览器与网页的兼容性问题。

4.1.1　Web 标准概述

无规矩不成方圆，对于网页设计也是如此。为了 Web 更好地发展，对于开发人员和最终用户而言，非常重要的事情就是在开发新的应用程序时，浏览器开发商和站点开发商需要遵守同一标准，就是 Web 标准。

为了缩短开发和维护时间，未来的网站将不得不根据标准来进行编码。这样，开发人员就不必为了得到相同的结果，而挣扎于多版本的开发困扰中。一旦所有 Web 开发人员遵守了 Web 标准，那么开发人员就可以更容易地理解彼此的编码了，Web 开发团队所有协作也将会得到简化。

使用 Web 标准有以下几个优点。

1. 对于访问者

- 文件下载与页面显示速度更快。
- 内容能被更多的用户所访问（包括失明、视弱、色盲等残障人士）。
- 内容能被更广泛的设备所访问（包括屏幕阅读机、手持设备、打印机等等）。
- 用户能够通过样式选择定制自己的表现界面。
- 所有页面都能提供适于打印的版本。

2. 对于网站所有者

- 更少的代码和组件，容易维护。
- 带宽要求降低（代码更简洁），成本降低。
- 更容易被搜寻引擎搜索到。
- 改版方便，不需要变动页面内容。

- 提供打印版本而不需要复制内容。
- 提高网站易用性。在美国，有严格的法律条款（Section 508）来约束政府网站必须达到一定的易用性，其他国家也有类似的要求。

4.1.2 Web 标准规定的内容

WEB 标准不是某一个标准，而是一系列标准的集合。网页主要由三部分组成：结构（Structure）、表现（Presentation）和行为（Behavior），那么，对应的标准也分三方面，分别如下：

- 结构化标准语言主要包括 XHTML 和 XML。
- 表现标准语言主要包括 CSS。
- 行为标准主要包括对象模型，如 W3C DOM、ECMAScript 等。

这些标准大部分由 W3C 起草和发布，也有一些是其他标准组织制订的标准，比如 ECMA（European Computer Manufacturers Association）的 ECMAScript 标准。

1.结构标准语言

（1）XML

XML 是 Extensible Markup Language（可扩展标识语言）的简写。目前推荐遵循的是 W3C 于 2000 年 10 月 6 日发布的 XML1.0。和 HTML 一样，XML 同样来源于 SGML，但 XML 是一种能定义其他语言的语言。XML 最初设计的目的是弥补 HTML 的不足，以其强大的扩展性满足网络信息发布的需要，后来逐渐用于网络数据的转换和描述。

（2）XHTML

XHTML 是 Extensible HyperText Markup Language（可扩展超文本标识语言）的缩写。目前推荐遵循的是 W3C 于 2000 年 1 月 26 日发布的 XML1.0。XML 虽然数据转换能力强大，完全可以替代 HTML，但面对成千上万已有的站点，直接采用 XML 还为时过早。因此，在 HTML 4.0 的基础上，用 XML 的规则对其进行扩展，得到了 XHTML。简单的说，建立 XHTML 的目的就是实现 HTML 向 XML 的过渡。

2. 表现标准语言

CSS 是 Cascading Style Sheets（层叠样式表）的缩写。目前推荐遵循的是 W3C 于 1998 年 5 月 12 日分布的 CSS2。W3C 创建 CSS 标准的目的是以 CSS 取代 HTML 表格式布局、帧和其他表现的语言。纯 CSS 布局与结构式 XHTML 相结合能帮助设计师分离外观与结构，使站点的访问及维护更加容易。

3. 行为标准

（1）DOM

DOM 是 Document Object Model（文档对象模型）的缩写。根据 W3C DOM 规范，DOM

是一种与浏览器平台语言的接口，使得它可以访问页面其他的标准组件。简单理解，DOM 解决了 Netscaped 的 JavaScript 和 Microsoft 的 JScript 之间的冲突，给 Web 设计师和开发者提供标准的方法，让他们可以访问自己站点中的数据、脚本和表现层对像。

（2）ECMAScript

ECMAScript 是 ECMA（European Computer Manufacturers Association）制定的标准脚本语言（JavaScript）。目前推荐遵循的是 ECMAScript 262。

4.2　HTML 5 文档的基本结构

HTML 5 文档最基本的结构主要包括文档类型说明、开始标记、元信息、主体标记和页面注释标记等。

4.2.1　HTML 5 结构

在一个 HTML 文档中，必须包含<HTML></HMTL>标记，并且放在 HTML 文档中开始和结束位置。即每个文档以<HTML>开始，以</HTML>结束。

<HTML></HMTL>之间通常包含两个部分，分别是<head></head>和<body></body>，<head>标记包含 HTML 头部信息，例如文档标题、样式定义等。<body>包含文档主体部分，即网页内容。需要注意的是，HTML 标记不区分大小写。

为了便于读者从整体把握 HTML 文档结构，通过一个 HTML 页面来介绍 HTML 页面的整体结构，示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <title>网页标题</title>
</head>
<body>
    网页内容
</body>
</html>
```

从上面代码可以看出，一个基本的 HTML 页有以下几部分构成。

- <!DOCTYPE>：声明必须位于 HTML 5 文档中的第一行，也就是位于<html>标记之前。该标记告知浏览器文档所使用的 HTML 规范。<!DOCTYPE>声明不属于 HTML 标记；它是一条指令，告诉浏览器编写页面所用标记的版本。由于 HTML 5 版本还没有得到浏览器的完全认可，后面介绍时还采用以前通用的标准。
- <html></html>：说明本页面使用 HTML 语言编写，使浏览器软件能够准确无误地解释、显示。

- <head></head>：是 HTML 的头部标记，头部信息不显示在网页中，此标记可以保护其他标记。用于说明文件标题和整个文件的一些公用属性。可以通过<style>标记定义 CSS 样式表，通过<script>标记定义 JavaScript 脚本文件。
- <title></title>：是 head 中的重要组成部分，它包含的内容显示在浏览器的窗口标题栏中。如果没有 title，浏览器标题栏显示本页的文件名。
- <body></body>：包含 HTML 页面的实际内容，显示在浏览器窗口的客户区中。例如页面中文字、图像、动画、超链接以及其他 HTML 相关的内容都是在 body 标记里的定义。

4.2.2 文档类型说明

Web 页面的文档类型说明（DOCTYPE）被极大地简化了。细心的读者会发现，在第 1 章中使用 Dreamweaver CS5.5 创建的 HTML 文档时，文档头部的类型说明代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

上面为 XHTML 文档类型说明，读者可以看到这段代码既麻烦又难记，HTML 5 对文档类型进行了简化，简单到 15 个字符就可以了，代码如下：

```
<!DOCTYPE html>
```


注 意

文档类型的申明需要出现在 HTML 5 文件的第一行。

4.2.3 HTML 5 标记

HTML 5 标记代表文档的开始，由于 HTML 5 语言语法的松散特性，该标记可以省略，但是为了使之符合 Web 标准和文档的完整性，养成良好的编写习惯，建议不要省略该标记。

HTML 5 标记以<html>开头，以</html>结尾，文档的所有内容书写在开头和结尾的中间部分。语法格式如下：

```
<html>
...
</html>
```

4.2.4 头标记

头标记 head 用于说明文档头部相关信息，一般包括标题信息、元信息、定义 CSS 样式和脚本代码等。HTML 的头部信息以<head>开始，以</head>结束，语法格式如下：

```
<head>
…
```

```
</head>
```

<head>元素的作用范围是整篇文档，定义在 HTML 语言头部的内容往往不会在网页上直接显示。

1．标题标记

HTML 页面的标题一般是用来说明页面的用途，它显示在浏览器的标题栏中。在 HTML 文档中，标题信息设置在<head>与</head>之间。标题标记以<title>开始，以</title>结束，语法格式如下：

```
<title>
…
</title>
```

在标记中间的“… ”就是标题的内容，它可以帮助用户更好地识别页面。预览网页时，设置的标题在浏览器的左上方标题栏中显示，此外，在 Windows 任务栏中显示的也是这个标题，如图 4-1 所示。



图 4-1　标题栏在浏览器中的显示效果



页面的标题只有一个，在 HTML 文档的头部，即<head>和</head>之间。

2．元信息标记

<meta>元素可提供有关页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词。

<meta>标签位于文档的头部，不包含任何内容。<meta>标签的属性定义了与文档相关联

的名称/值对，<meta>标签提供的属性及取值，如下表所示。

属性	值	描述
charset	character encoding	定义文档的字符编码
content	some_text	定义与 http-equiv 或 name 属性相关的元信息
http-equiv	content-type expires refresh set-cookie	把 content 属性关联到 HTTP 头部
name	author description keywords generator revised others	把 content 属性关联到一个名称

（1）字符集 charset 属性

在 HTML 5 中，有一个新的 charset 属性，它使字符集的定义更加容易。例如，下列代码告诉浏览器，网页使用“ISO-8859-1”字符集显示，代码如下：

```
<meta charset="ISO-8859-1">
```

（2）搜索引擎的关键字

在早期，Meta Keywords 关键字对搜索引擎的排名算法起到一定的作用，也是很多人进行网页优化的基础。关键字在浏览时是看不到的，使用格式如下：

```
<meta name="keywords" content="关键字,keywords" />
```

说明：

- 不同的关键词之间应用半角逗号隔开（英文输入状态下），不要使用“空格”或“|”间隔;
- 其中，关键字是 keywords，不是 keyword;
- 关键字标签中的内容应该是一个个的短语，而不是一段话。

例如，定义针对搜索引擎的关键词，代码如下：

```
<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript" />
```

关键字标签 keywords，曾经是搜索引擎排名中很重要的因素，但现在已经被很多搜索引擎完全忽略。如果加上这个标签对网页的综合表现没有坏处，不过，如果使用不恰当的话，对网页非但没有好处，还有欺诈的嫌疑。在使用关键字标签 keywords 时，要注意以下几点：

- 关键字标签中的内容要与网页核心内容相关，确信使用的关键字出现在网页文本中。
- 使用用户易于通过搜索引擎检索的关键字，过于生僻的词汇不太适合做 meta 标签中的

关键词。

- 不要重复使用关键词，否则可能会被搜索引擎惩罚。
- 一个网页的关键词标签里最多包含 3~5 个最重要的关键词，不要超过 5 个。
- 每个网页的关键词应该不一样。


注 意

由于设计者或 SEO 优化者以前对 meta keywords 关键字的滥用，导致目前它在搜索引擎排名中的作用很小。

（3）页面描述

meta description 元标签（描述元标签）是一种 HTML 元标签，用来简略描述网页的主要内容，通常被搜索引擎用在搜索结果页上给最终用户看的一段文字片段。页面描述在网页中是不显示出来的，页面描述的使用格式如下：

```
<meta name="description" content="网页的介绍" />
```

例如，定义对页面的描述，代码如下：

```
<meta name="description" content="免费的 web 技术教程。" />
```

（4）页面定时跳转

使用<meta>标记可以使网页经过一定时间后自动刷新，这可通过将 http-equiv 属性值设置为 refresh 来实现。content 属性值可以设置为更新时间。

在浏览网页时经常会看到一些欢迎信息的页面，在经过一段时间后，这些页面会自动转到其他页面，这就是网页的跳转。页面定时刷新跳转的语法格式如下：

```
<meta http-equiv="refresh" content="秒;[url=网址]" />
```

说明：上面的[url=网址]部分是可选项，如果有这部分，页面定时刷新并跳转，如果省略该部分，页面只定时刷新，不进行跳转。

例如，实现每 5 秒刷新一次页面，将下述代码放入 head 标记部分即可：

```
<meta http-equiv="refresh" content="5" />
```

4.2.5 网页的主体标记

网页所要显示的内容都放在网页的主体标记内，它是 HTML 文件的重点所在。主体标记是以<body>开始，以</body>标记结束，语法格式如下：

```
<body>
…
</body>
```

注意，在构建 HTML 结构时，标记不允许交错出现，否则会造成错误。

例如，在下列代码中，<body>开始标记出现在<head>标记内：

```
<html>
<head>
<title>标记测试</title>
<body>
</head>
</body>
</html>
```

代码中的第 4 行<body>开始标记和第 5 行的</head>结束标记出现了交叉，这是错误的。HTML 中所有代码都不允许交错出现。

4.2.6　页面注释标记

注释是在 HTML 代码中插入的描述性文本，用来解释该代码或提示其他信息。注释只出现在代码中，浏览器对注释代码不进行解释，并且在浏览器的页面中不显示。

在 HTML 源代码中适当的地插入注释语句是一种非常好的习惯。对于设计者日后的代码修改、维护工作很有好处。另外，如果将代码交给其他设计者，其他人也能很快读懂前者所撰写的内容。

语法：

```
<!--注释的内容-->
```

注释语句元素由前后两半部分组成，前半部分一个左尖括号、一个半角感叹号和两个连字符头，后半部分由两个连字符和一个右尖括号组成。

```
<html>
<head>
<title>标记测试</title>
</head>
<body>
<!-- 这里是标题-->
<h1>HTML 5 从入门到精通</h1>
</body>
</html>
```

页面注释不但可以对 HTML 中一行或多行代码进行解释说明，而且可能注释掉这些代码。如果希望某些 HTML 代码在浏览器中不显示，可以将这部分内容放在“<!—”和“-->”之间，例如，修改上述代码，如下所示：

```
<html>
<head>
<title>标记测试</title>
```

```
</head>
<body>
<!—
<h1>HTML 5 从入门到精通</h1>
-->
</body>
</html>
```

修改后的代码，将<h1>标记作为注释内容处理，在浏览器中将不会显示这部分内容。

4.3 综合实例——符合 W3C 标准的 HTML 5 网页

下面制作一个简单的符合 W3C 标准的 HTML 5 网页，以巩固前面所学知识。

具体操作步骤如下：

01 启动 Dreamweaver CS5.5，新建 HTML 文档，并切单击文档工具栏中的“代码”视图按钮，切换至代码状态，如图 4-2 所示。



图 4-2 使用 Dreamweaver CS5.5 新建 HTML 文档

02 图 4-2 中的代码是 XHTML 1.0 格式，尽管与 HTML 5 完全兼容，但是为了简化代码，现在将其修改成 HTML 5 规范。修改文档说明部分、<html>标记部分和<meta>元信息部分，修改后，HTML 5 基本结构代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>HTML 5 网页设计</title>
</head>

<body>
</body>
</html>
```

03 在网页主体中添加内容，在 body 部分增加如下代码：

```
<!--白居易诗-->
<h1>续座右铭</h1>
<P>
 千里始足下,<br>
 高山起微尘。<br>
 吾道亦如此,<br>
 行之贵日新。<br>
</P>
```

04 保存网页，在 IE 9.0 中的预览效果，如图 4-3 所示。



图 4-3　网页预览效果

4.4　问题解答

1. 在网页中，语言的编码方式有哪些？

在 HTML 5 网页中，<meta>标记的 charset 属性用于设置网页的内码语系，也就是字符集的类型，中国常用的是 GB 码，因为在中国要经常显示汉字，通常设置为 GB2312（简体中文）和 UTF-8 两种。英文是 ISO-8859-1 字符集，此外还有其他的字符集，这里不再介绍。

2. 在网页中基本标签是否必须成对出现？

在 HTML 5 网页中，大部分标签都成对出现，不过也有部分标签可以单独出现。例如换行标签<p/>、
、<img/>和<hr/>等。

第 5 章　HTML 5 中的文本和图像

文字和图像是网页中最主要也是最常用的元素。因此在这一章，开始讲解如何在网页中使用文字、文字结构标记以及图像。

5.1　添加文本

网页中的文本可以分为两大类：一类是普通文本，另外一类是特殊字符文本。

5.1.1　普通文本

所谓普通文本是指汉字或者在键盘上可以输出的字符。读者可以在 Dreamweaver CS5.5 代码视图的 body 标签部分直接输入；或者在设计视图下直接输入。

如果有现成的文本，可以使用复制、粘贴的方法把其他窗口中需要的文本复制过来。在粘贴文本的时候，如果只希望把文字粘贴过来，而不需要粘贴其他文档中的格式，可以使用 Dreamweaver CS5.5 的“选择性粘贴”功能。

“选择性粘贴”功能只在 Dreamweaver CS5.5 的设计视图中起作用，因为在代码视图中，粘贴的是“仅文本”，不会有格式。例如，将 Word 文档表格中的文字复制到网页中，而不需要表格结构。操作方法：选择“编辑”→“选择性粘贴…”或按下快捷键 Ctrl+Shift+V，弹出“选择性粘贴”对话框，在对话框中选择“仅文本”选项，如图 5-1 所示。



图 5-1　“选择性粘贴”对话框

5.1.2　特殊字符文本

目前，很多行业信息都出现在网络上，每个行业都有自己的行业特性，如数学、物理和化学都有特殊的符号。如何在网页上显示这些特殊符号是这节为读者讲述的内容。

在 HTML 中，特殊符号以“&”符号开头，后面跟相关字符。例如，大括号和小括号被用于声明标记，因此如果在 HTML 代码中出现“<”和“>”字符，就不能直接输入了，需要当作特殊字符处理。HTML 中，用“&lt”代表符号“<”，用“&gt”代表符号“>”。如，输入公式 a>b，在 HTML 中需要这样表示：a>b。

HTML 中还有大量这样的字符，例如，空格、版权等，常用特殊字符如下表所示。

显示	说明	HTML 编码
	半角大的空白	
	全角大的空白	
	不断行的空白格	
<	小于	<
>	大于	>
&	&符号	&
"	双引号	"
©	版权	©
®	已注册商标	®
™	商标（美国）	™
×	乘号	×
÷	除号	÷

在编辑化学公式或物理公式时，使用特殊字符的频率非常高。如果每次输入时都去查询或者要记忆这些特殊符号的编码，工作量是相当大的。在此作者为读者提供一些技巧。

（1）在 Dreamweaver CS5.5 的设计视图下输入字符，如输入 a>b 这样的表达式，可以直接输入。对于部分键盘上没有的字符可以借助“中文输入法”的软键盘。在中文输入法的软键盘上单击鼠标右键，弹出特殊类别项，如图 5-2 所示。选择所需类型，如选择“数学符号”，弹出数学相关符号，如图 5-3 所示。单击“÷”号，即可输入。

P C 键盘	标点符号
希腊字母	数字序号
俄文字母	数学符号
注音符号	单位符号
拼　　音	制表符
日文平假名	特殊符号
日文片假名	

中 极品五笔

图 5-2　特殊符号分类



图 5-3　数学符号

（2）文字与文字之间的空格，如果超过一个，那么从第 2 个空格开始，都会被忽略掉。快捷的输入空格的方法如下：将输入法切换成“中文输入法”，并置于“全角”（Shift+空格）

状态，直接单击键盘上的空格键即可。

（3）对于上述两种方法都无法实现的字符，可以使用 Dreamweaver CS5.5 的“插入”菜单实现。选择“插入”→“HTML”→“特殊字符”菜单命令，在所需要的字符中选择，如果没有，选择“其他字符…”选项。



尽量不要使用多个“ ”来表示多个空格，因为多数浏览器对空格的距离的实现是不一样的。

5.1.3 文本特殊样式

在文档中经常会出现重要文本（加粗显示）、斜体文本、上标和下标文本等。

1. 重要文本

重要文本通常以粗体显示、强调方式显示或加强调方式显示。HTML 中的<b>标记、<em>标记和<strong>标记分别实现了这三种显示方式。

【例 5.1】（实例文件：ch05\5.1.html）

```
<!DOCTYPE html>
<html>
<head>
<title>无标题文档</title>
</head>
<body>
<p><b>我是粗体文字</b></p>
<p><em>我是强调文字</em></p>
<p><strong>我是加强调文字</strong></p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-4 所示，实现了文本的三种显示方式。

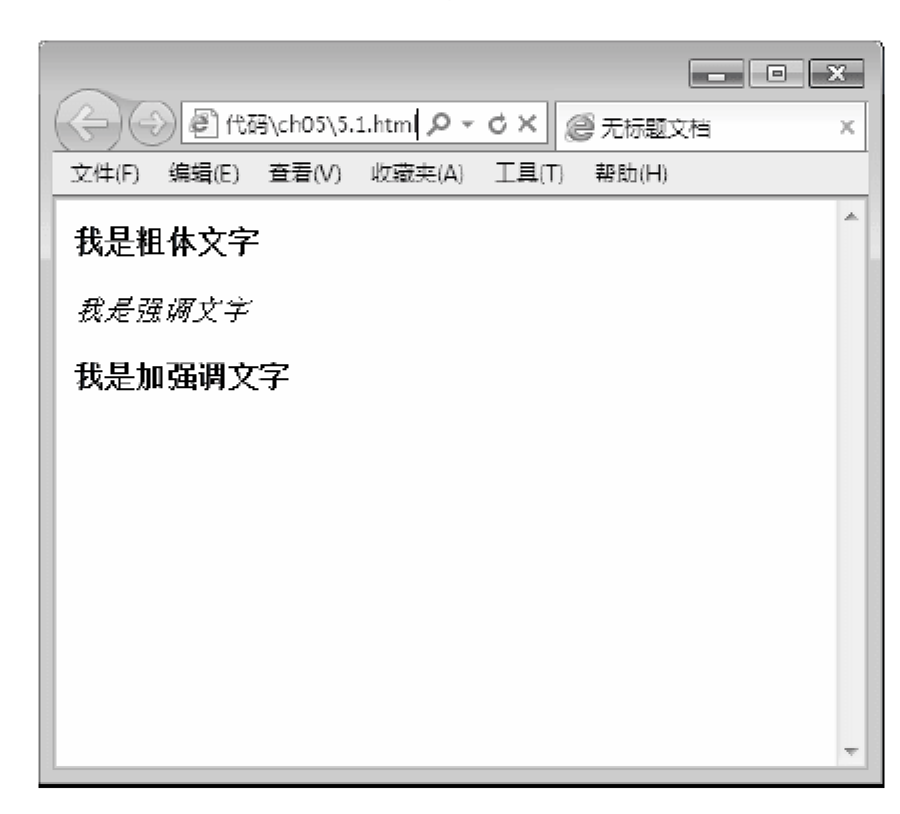


图 5-4 重要文本预览效果

2．斜体文本

HTML 中的<i>标记实现了文本的斜体显示。放在“<i>”、“</i>”之间的文本将以斜体显示。

【例 5.2】（实例文件：ch05\5.2.html）

```
<!DOCTYPE html>
<html>
<head>
<title>无标题文档</title>
</head>
<body>
<i>我将会以斜体字显示</i>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-5 所示，其中文字以斜体显示。



图 5-5 斜体文本预览效果



HTML 中的“重要文本”和斜体文本标记已经过时，可使用 CSS 样式来实现，而不是使用 HTML 来实现。随着后面的学习，读者会发现，即使 HTML 和 CSS 实现相同的效果，但是 CSS 所能实现的控制远比 HTML 要细致、精确很多。

3．上标和下标文本

在 HTML 中用<sup>标记实现上标文字，用<sub>标记实现下标文字。<sup>和<sub>都是双标记，放在开始标记和结束标记之间的文本会分别以上标或下标形式出现。

【例 5.3】（实例文件：ch05\5.3.html）

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>无标题文档</title>
</head>
<body>
<!-上标显示-->
<p>c=a<sup>2</sup>+b<sup>2</sup></p>
<!-下标显示-->
<p>H<sub>2</sub>+O→H<sub>2</sub>O</p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-6 所示，分别实现了上标和下标文本显示。

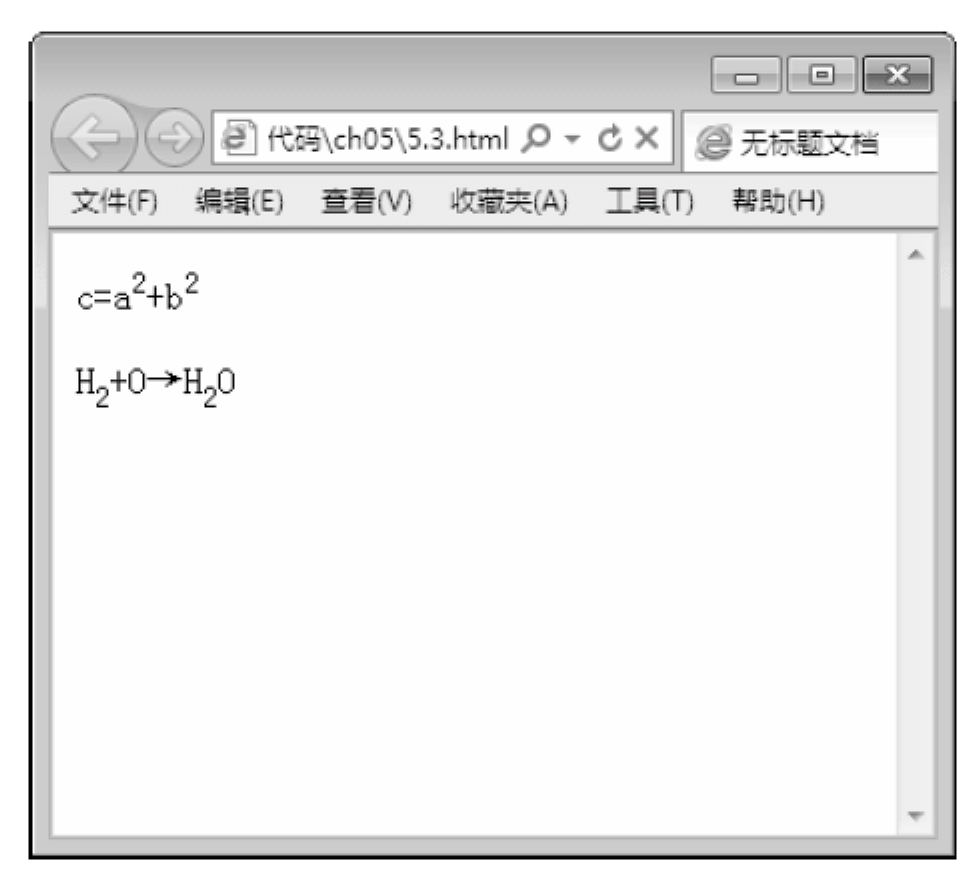


图 5-6　上标和下标预览效果

5.2　排版文本

在网页中如果要把文字合理地显示出来，离不开段落标记的使用。对网页中文字段落进行排版，并不像文本编辑软件 Word 那样可以定义许多模式来安排文字的位置。在网页中要让某一段文字放在特定的地方是通过 HTML 标记来完成的。

5.2.1　段落标记与换行标记

浏览器在显示网页时，完全按照 HTML 标记来解释 HTML 代码，忽略多余的空格和换行。在 HTML 文件里，不管输入多少空格（按空格键）都将被视为一个空格；换行（按 Enter 键）也是无效的。在 HTML 中，换行使用
标记，段落使用<p>标记。

1．换行标记

换行标记
是一个单标记，它没有结束标记，是英文单词 break 的缩写，作用是将文字在一个段内强制换行。一个
标记代表一个换行，连续的多个标记可以实现多次换行。使用换行标记时，在需要换行的位置添加
标记即可。例如下面的代码，实现了对文本的强制换行。

【例 5.4】（实例文件：ch05\5.4.html）

```
<!DOCTYPE html>
<html>
<head>
<title>文本段换行</title>
</head>
<body>
本节目标<br/>
网页中的文字是如何设置的<br/>
如何在 Dreamweaver 中处理文字<br/>
如何对文本进行格式化（CSS）<br/>
熟悉使用 Dreamweaver 进行样式表的创建与应用
</body>
</html>
```

虽然在 HTML 源代码中，主体部分的内容在排版上没有换行，但是增加
标记后，在 IE 9.0 中的预览效果如图 5-7 所示，实现了换行效果。

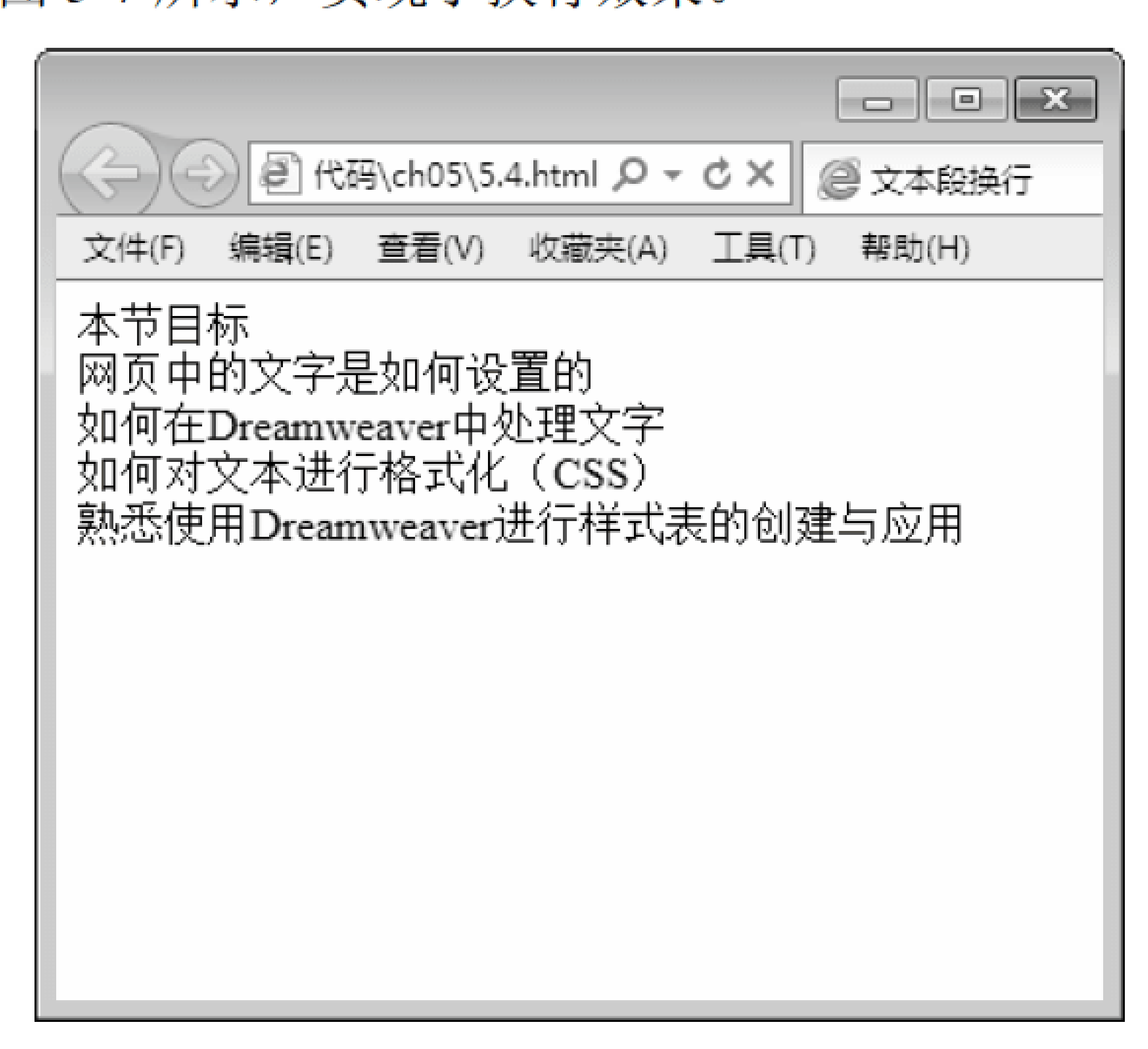


图 5-7　换行标记的效果

2. 段落标记<p>

段落标记是双标记，即<p></p>，在<p>开始标记和</p>结束标记之间的内容形成一个段落。如果省略结束标记，那么从<p>标记开始，直到遇见下一个段落标记之前的文本，都在一段段落内。段落标记中的 p 是英文单词 paragraph 即“段落”的首字母，用来定义网页中的一段文本，文本在一个段落中会自动换行。

【例 5.5】（实例文件：ch05\5.5.html）

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>段落标记的使用</title>
</head>
<body>
<p>HTML 5、CSS3 应用教程之 跟 DIV 说 Bey!Bey!</p>
<p>Web 设计师可以使用 HTML4 和 CSS2.1 完成一些很酷的东西。我们可以在不使用陈旧的基于 table 布局的基础上完成文档逻辑结构并创建内容丰富的网站。我们可以在不使用内联<font>和<br>标签的基础上对网站添加漂亮而细腻的风格样式。事实上，我们目前的设计能力已经让我们远离了那个可怕的浏览器战争时代、专有协议和那些充满闪动、滚动和闪烁的丑陋网页。
<p>
<p>
虽然我们现在已经普遍使用了 HTML4 和 CSS2.1，但是我们还可以做得更好！我们可以重组我们代码的结构并能让我们的页面代码更富有语义化特性。我们可以缩减带给页面美丽外观样式代码量并让他们有更高的可扩展性。现在，HTML 5 和 CSS3 正跃跃欲试的等待大家，下面让我们来看看他们是否真的能让我们的设计提升到下一个高度吧…
</p>
<p>
曾经，设计师们经常会跟频繁使用基于 table 的没有任何语义的布局。不过最终还是要感谢像 Jeffrey Zeldman 和 Eric Meyer 这样的思想革新者，聪明的设计师们慢慢的接受了相对更语义化的<div>布局替代了 table 布局，并且开始调用外部样式表。但不幸的是，复杂的网页设计需要大量不同的标签结构代码，我们把它叫做“<div>-soup” 综合症。
</p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-8 所示，<P>标记将文本分成 4 个段落。

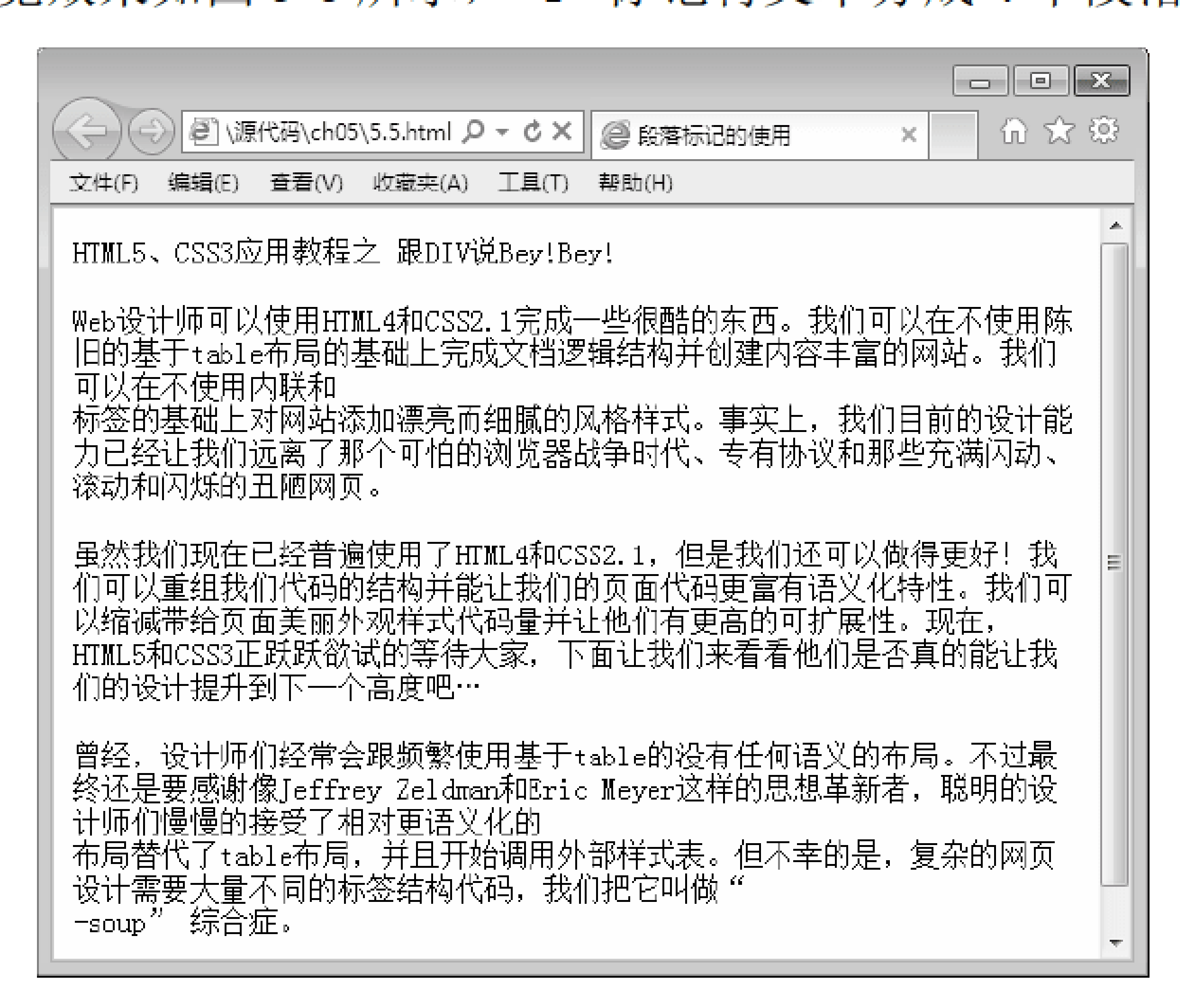


图 5-8 段落标记的使用

5.2.2　标题标记

在 HTML 文档中，文本的结构除了以行和段出现之外，还可以作为标题存在。通常一篇文档最基本的结构就是由若干不同级别的标题和正文组成的。

HTML 文档中包含有各种级别的标题，各种级别的标题由<h1>到<h6>元素来定义，<h1>至<h6>标题标记中的字母 h 是英文 headline（标题行）的简称。其中<h1>代表 1 级标题，级别最高，文字也最大，其他标题元素依次递减，<h6>级别最低。

【例 5.6】（实例文件：ch05\5.6.html）

```
<!DOCTYPE html>
<html>
<head>
<title>文本段换行</title>
</head>
<body>
<h1>这里是 1 级标题</h1>
<h2>这里是 2 级标题</h2>
<h3>这里是 3 级标题</h3>
<h4>这里是 4 级标题</h4>
<h5>这里是 5 级标题</h5>
<h6>这里是 6 级标题</h6>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-9 所示。

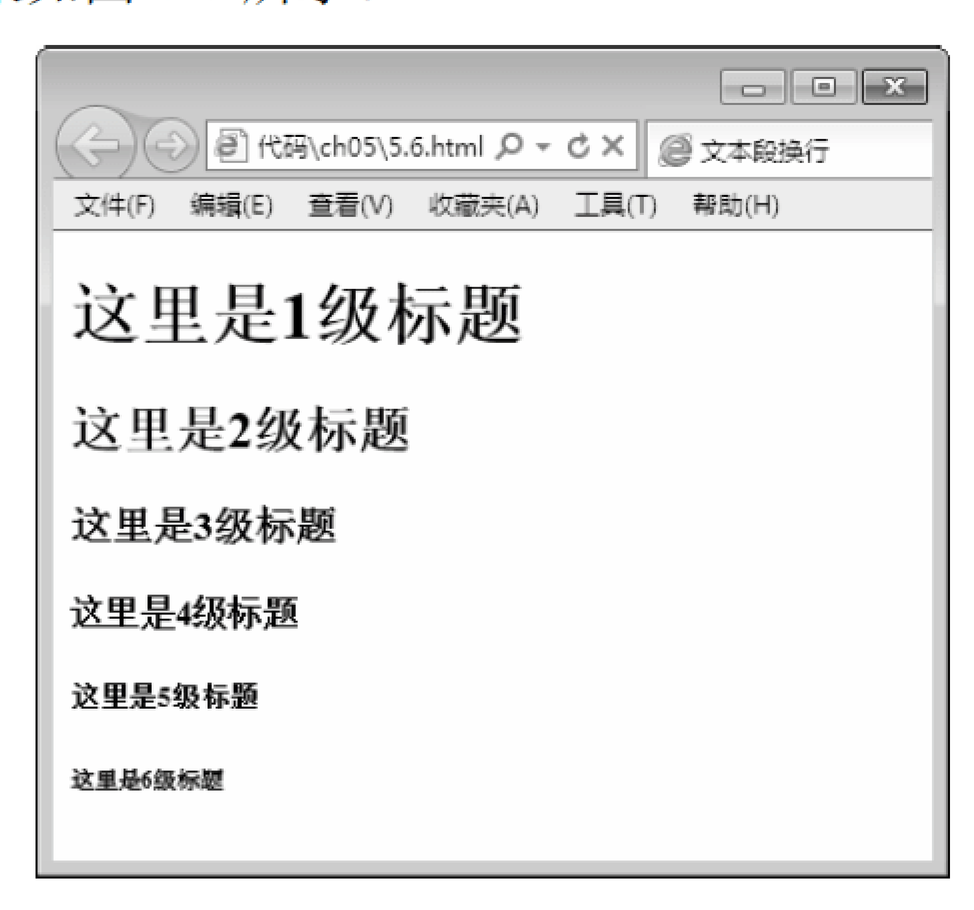


图 5-9　标题标记的使用



作为标题，它们的重要性是有区别的，其中<h1>标题的重要性最高，<h6>的最低。

5.3　建立文本列表

使用文本列表可以有序地编排一些信息资源，使其结构化和条理化，并以列表的样式显示出来，以便浏览者能更加快捷地获得相应信息。HTML 中的文字列表如同文字编辑软件 Word 中的项目符号和自动编号。

5.3.1　建立无序列表

无序列表相当于 Word 中的项目符号，无序列表的项目排列没有顺序，只以符号作为分项标识。无序列表使用一对标记
，其中每一个列表项使用
，其结构如下所示。

```
<ul>
  <li>无序列表项</li>
  <li>无序列表项</li>
  <li>无序列表项</li>
  <li>无序列表项</li>
</ul>
```

在无序列表结构中，使用
标记表示这个无序列表的开始和结束，<li>则表示一个列表项的开始。在一个无序列表中可以包含多个列表项，并且<li>可以省略结束标记。下面实例使用无序列表实现文本的排列显示。

【例 5.7】（实例文件：ch05\5.7.html）

```
<!DOCTYPE html>
<html>
<head>
<title>嵌套无序列表的使用</title>
</head>

<body>
<h1>网站建设流程</h1>
<ul>
    <li>项目需求</li>
    <li>系统分析
      <ul>
        <li>网站的定位</li>
        <li>内容收集</li>
        <li>栏目规划</li>
        <li>网站目录结构设计</li>
        <li>网站标志设计</li>
        <li>网站风格设计</li>
        <li>网站导航系统设计</li>
      </ul>
```

```
    </li>
    <li>伪网页草图
        <ul>
          <li>制作网页草图</li>
          <li>将草图转换为网页</li>
        </ul>
    </li>
    <li>站点建设</li>
    <li>网页布局</li>
    <li>网站测试</li>
    <li>站点的发布与站点管理</li>
</ul>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-10 所示。读者会发现，无序列表项中，可以嵌套一个列表。如代码中的“系统分析”列表项和“伪网页草图”列表项中都有下级列表，因此在这对
标记间又增加了一对
标记。

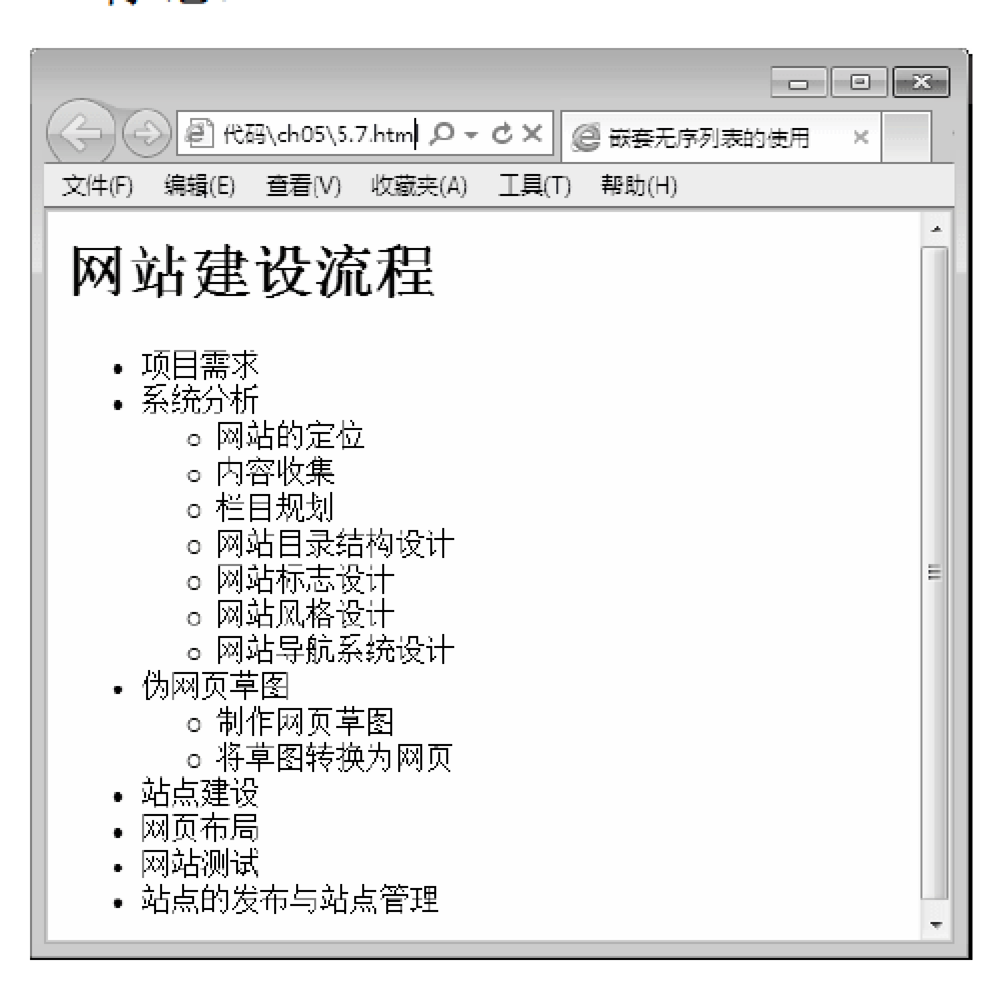


图 5-10 无序列表

5.3.2 建立有序列表

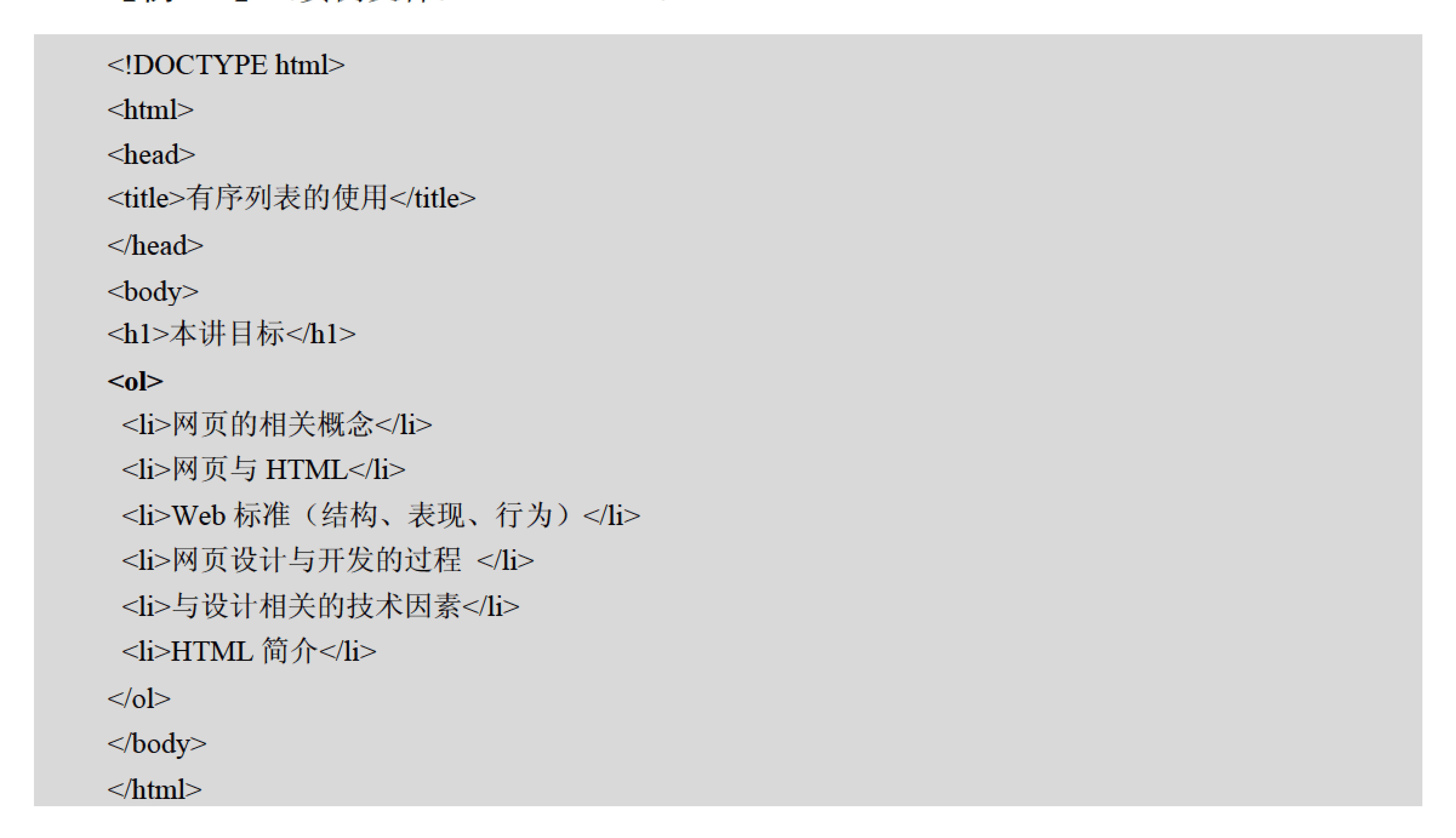
有序列表类似于 Word 中的自动编号功能，有序列表的使用方法和无序列表的使用方法基本相同，它使用标记
，每一个列表项前使用
。每个项目都有前后顺序之分，多数用数字表示，其结构如下：

```
<ol>
  <li>第 1 项</li>
  <li>第 2 项</li>
  <li>第 3 项</li>
```

```
</ol>
```

下面实例使用有序列表实现文本的排列显示。

【例 5.8】（实例文件：ch05\5.8.html）



```
<!DOCTYPE html>
<html>
<head>
<title>有序列表的使用</title>
</head>
<body>
<h1>本讲目标</h1>
<ol>
  <li>网页的相关概念</li>
  <li>网页与 HTML</li>
  <li>Web 标准（结构、表现、行为）</li>
  <li>网页设计与开发的过程 </li>
  <li>与设计相关的技术因素</li>
  <li>HTML 简介</li>
</ol>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-11 所示。用户可以看到新添加的有序列表。

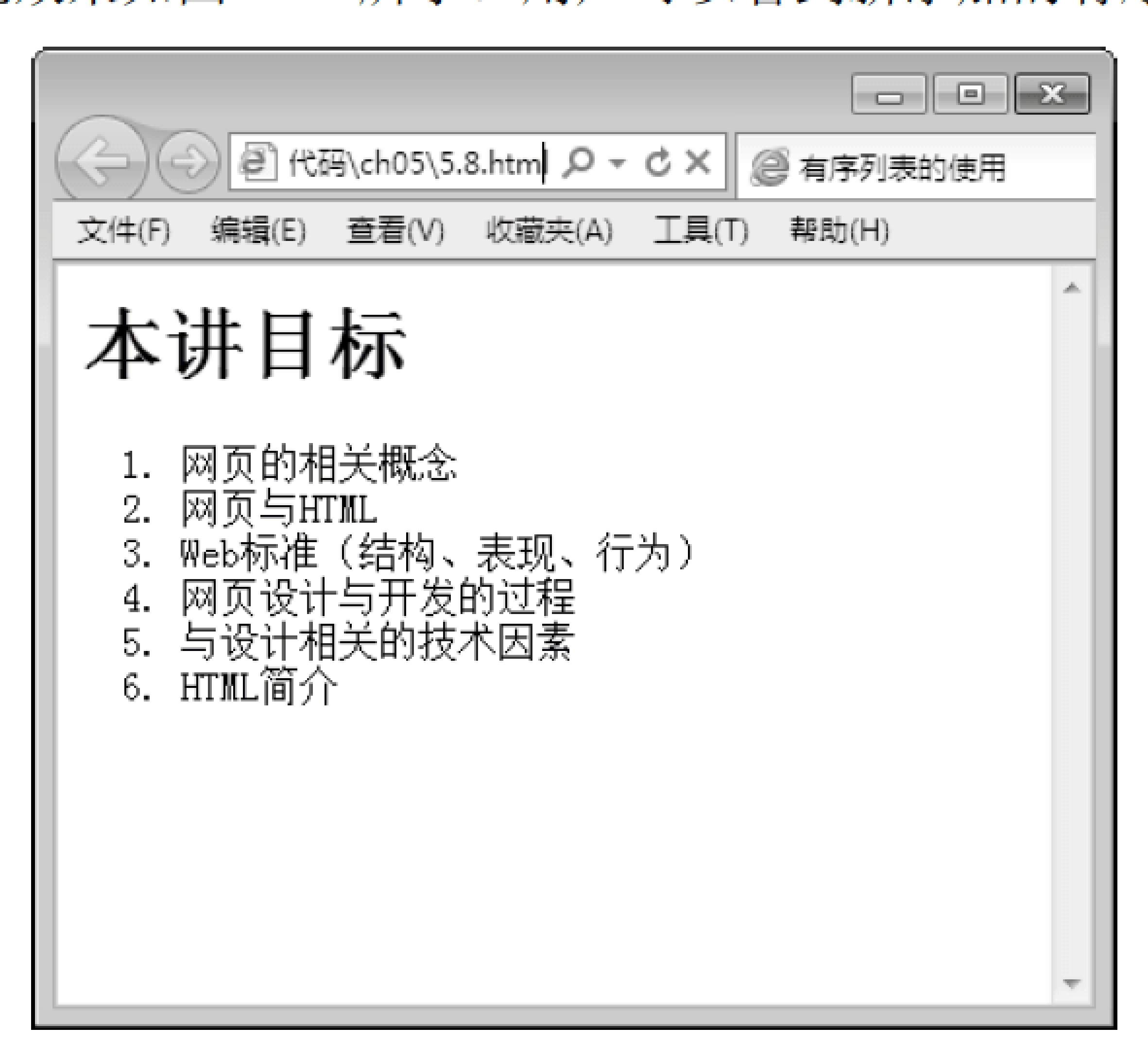


图 5-11　有序列表的效果

5.3.3　建立不同类型的无序列表

通过使用多个<ul>标签，可以建立不同类型的无序列表。

【例 5.9】（实例文件：ch05\5.9.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>Disc 项目符号列表：</h4>
<ul type="disc">
<li>苹果</li>
<li>香蕉</li>
<li>柠檬</li>
<li>桔子</li>
</ul>
<h4>Circle 项目符号列表：</h4>
<ul type="circle">
<li>苹果</li>
<li>香蕉</li>
<li>柠檬</li>
<li>桔子</li>
</ul>
<h4>Square 项目符号列表：</h4>
<ul type="square">
<li>苹果</li>
<li>香蕉</li>
<li>柠檬</li>
<li>桔子</li>
</ul>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-12 所示。



图 5-12　不同类型的无序列表

5.3.4 建立不同类型的有序列表

通过使用多个<ol>标签，可以建立不同类型的有序列表。

【例 5.10】（实例文件：ch05\5.10.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>数字列表：</h4>
<ol>
<li>苹果</li>
<li>香蕉</li>
<li>柠檬</li>
<li>桔子</li>
</ol>
<h4>字母列表：</h4>
<ol type="A">
<li>苹果</li>
<li>香蕉</li>
<li>柠檬</li>
<li>桔子</li>
</ol>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-13 所示。

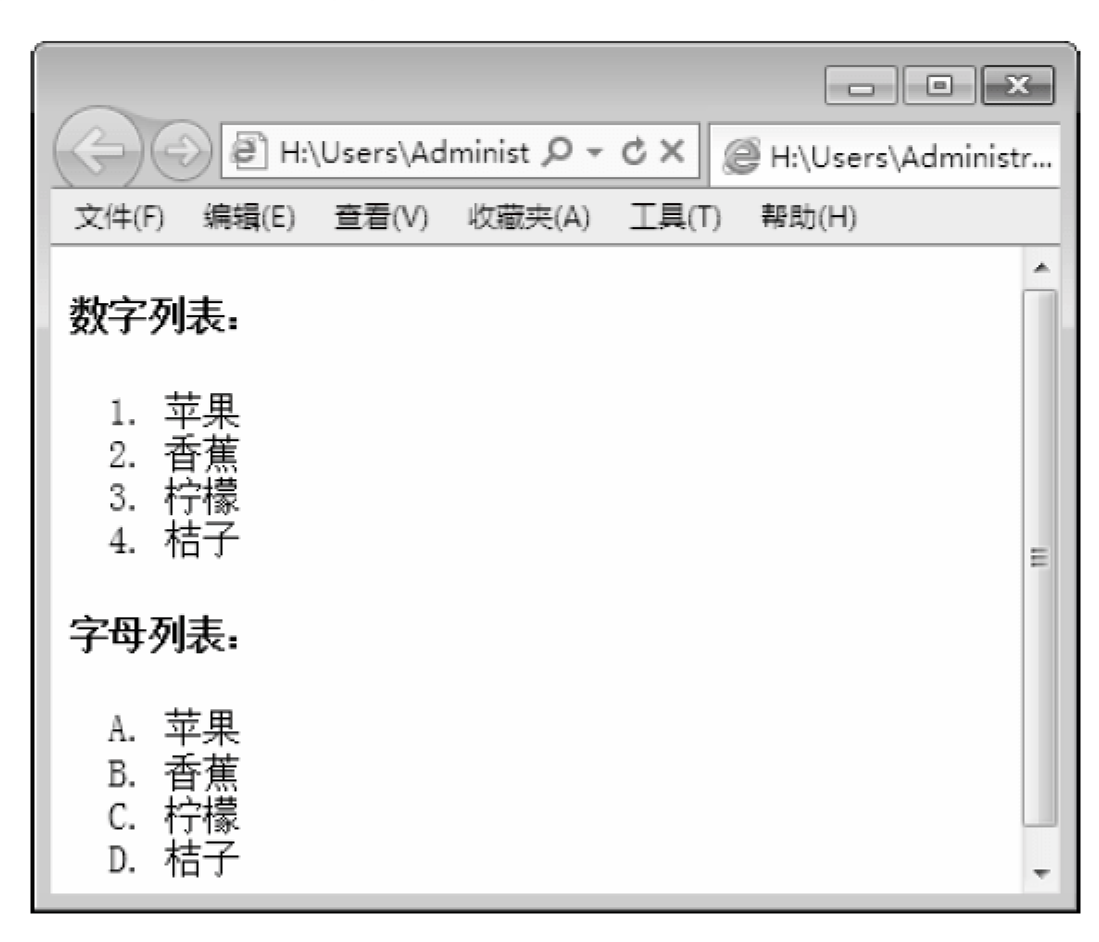


图 5-13 不同类型的有序列表

5.3.5 嵌套列表

嵌套列表是网页中常用的元素，使用<ul>标签可以制作网页中的嵌套列表。

【例 5.11】（实例文件：ch05\5.11.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>一个嵌套列表：</h4>
<ul>
 <li>咖啡</li>
 <li>茶
   <ul>
   <li>红茶</li>
   <li>绿茶
     <ul>
     <li>中国茶</li>
     <li>非洲茶</li>
     </ul>
   </li>
   </ul>
 </li>
 <li>牛奶</li>
</ul>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-14 所示。



图 5-14 嵌套列表

5.3.6 自定义列表

在 HTML 5 中还可以自定义列表，标签是<dl>。

【例 5.12】（实例文件：ch05\5.12.html）

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>一个定义列表：</h2>
<dl>
  <dt>电脑</dt>
  <dd>是一种能够按照程序运行的电子设备…….</dd>
  <dt>显示器</dt>
  <dd>以视觉方式显示信息的装置 ... ...</dd>
</dl>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-15 所示。

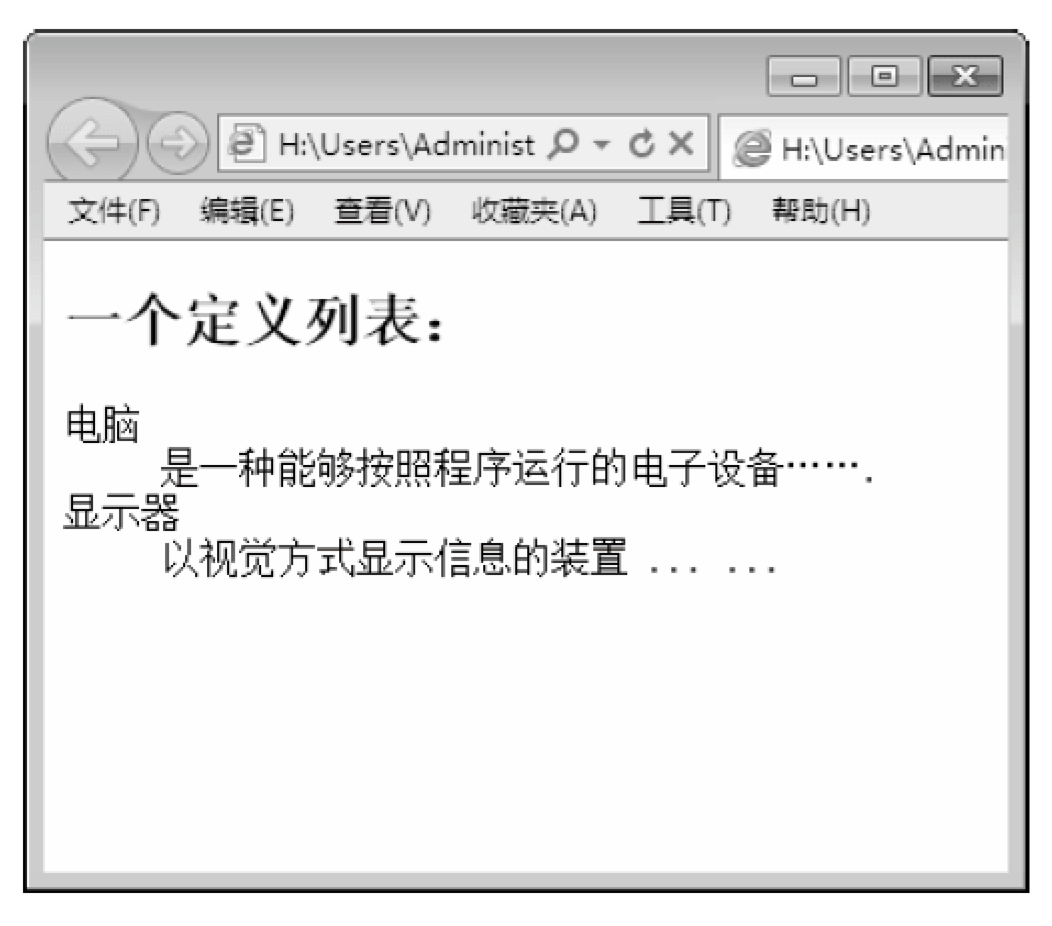


图 5-15 自定义列表

5.4 添加图像

俗话说“一图胜千言”，图片是网页中不可缺少的元素，巧妙地在网页中使用图片可以为网页增色不少。网页支持多种图片格式，并且可以对插入的图片设置宽度和高度。

5.4.1 网页支持的图片格式

图像在网页中具有画龙点睛的作用，它能装饰网页，表达个人的情调和风格。但在网页上加入的图片越多，浏览的速度就会受到影响。网页中使用的图像可以是 GIF、JPEG、BMP、TIFF、PNG 等格式的图像文件，其中使用最广泛的主要是 GIF 和 JPEG 两种格式。

1. GIF 格式

GIF 格式是由 Compuserve 公司提出的与设备无关的图像存储标准，也是 Web 上使用最早、应用最广的图像格式。GIF 是通过减少组成图像像素的存储位数和 LZH 压缩存储技术来减少图像文件的大小的，GIF 格式最多只能是 256 色的图像。

GIF 具有图像文件短小、下载速度快的特点，在低颜色数下 GIF 比 JPEG 装载更快，可用许多具有同样大小的图像文件组成动画，在 GIF 图像中可指定透明区域，使图像具有非同一般的显示效果。

2. JPEG 格式

JPEG 格式是在目前 Internet 中最受欢迎的图像格式，JPEG 可支持多达 16MB 的颜色，能展现十分生动的图像，还能压缩。但压缩方式是以损失图像质量为代价的，压缩比越高图像质量损失越大，图像文件也就越小。

而 Windows 支持的位图 BMP 格式的图像，一般情况下，同一图像的 BMP 格式的大小是 JPEG 格式的 5~10 倍。而 GIF 格式最多只能是 256 色，因此载入 256 色以上图像的 JPEG 格式成了 Internet 中最受欢迎的图像格式。

当网页中需要载入一个较大的 GIF 或 JPEG 图像文件时，装载速度会很慢。为改善网页的视觉效果，可在载入时设置为隔行扫描。隔行扫描在开始显示图像时看起来非常模糊，接着细节逐渐添加上去直到图像完全显示出来。


注 意

现在网页中也有很多 PNG 格式的图片。PNG 图片具有不失真、兼有 GIF 和 JPG 的色彩模式、网络传输速度快、支持透明图像制作的特点，近年来在网络中也很流行。

5.4.2　在网页中使用路径

HTML 文档支持文字、图片、声音、视频等媒体格式，但是在这些格式中，除了文本是写在 HTML 中的，其他都是嵌入式的，HTML 文档只记录了这些文件的路径。这些媒体信息能否正确显示，路径是至关重要的。

路径的作用是定位一个文件的位置。文件的路径可以有两种表述方法，以当前文档为参照物表示文件的位置，即相对路径。以根目录为参照物表示文件的位置，即绝对路径。

为了方便讲述绝对路径和相对路径，现有目录结构，如图 5-16 所示。

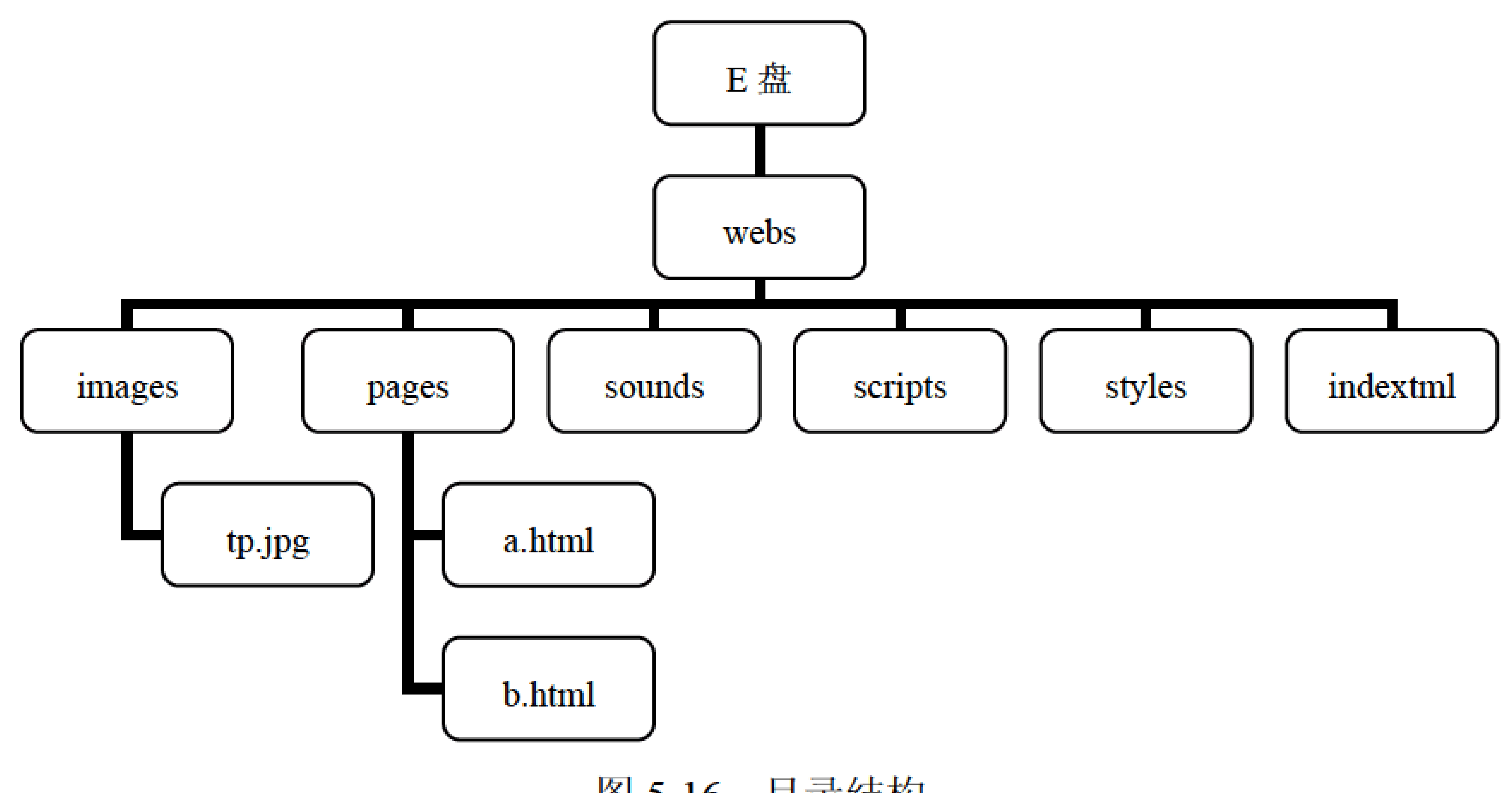


图 5-16　目录结构

1. 绝对路径

例如，在 E 盘的 webs 目录下的 images 下有一个 tp.jpg 图像，那么它的路径就是 E:\webs\images\tp.jpg，像这种完整地描述文件位置的路径就是绝对路径。如果将图片文件 tp.jpg 插入到网页 index.html，绝对路径表示方式如下：

```
E:\webs\images\tp.jpg
```

如果使用了绝对路径 E:\webs\images\tp.jpg 进行图片链接，那么在本地计算机中将一切正常，因为在 E:\webs\images 下的确存在 tp.jpg 这个图片。如果将文档上传到网站服务器上后，就不会正常了，因为服务器划分的存放空间可能在 E 盘其他目录中，也可能在 D 盘其他目录中。为了保证图片正常显示，必须从 webs 文件夹开始，放到服务器或其他计算机的 E 盘根目录下。

通过上述讲解，读者会发现，如果链接的资源是本站点内的，则使用绝对路径对位置要求非常严格。因此，链接本站内的资源时不建议采用绝对路径。如果链接其他站点的资源，则必须使用绝对路径。

2. 相对路径

如何使用相对路径设置上述图片呢？所谓相对路径，顾名思义就是以当前位置为参考点，自己相对于目标的位置。例如，在 index.html 中连接 tp.jpg 就可以使用相对路径。index.html 和 tp.jpg 图片的路径根据上述目录结构图可以这样来定位。从 index.html 位置出发，它和 images 属于同级，路径是通的，因此可以定位到 images，images 的下级就是 tp.jpg。使用相对路径表示图片如下：

```
images/tp.jpg
```

使用相对路径，不论将这些文件放到哪里，只要 tp.jpg 和 index.html 文件的相对关系没有变，就不会出错。

在相对路径中，“..”表示上一级目录，“../..”表示上级的上级目录，依此类推。例如，将 tp.jpg 图片插入到 a.html 文件中，使用相对路径表示如下：

```
../images/tp.jpg
```



细心的读者会发现，路径分隔符使用了“\”和“/”两种，其中“\”表示本地分隔符，“/”表示网络分隔符。因为网站制作好后肯定是在网络上运行，因此要求使用“/”作为路径分隔符。

有的读者可能会有这样的疑惑：一个网站有许多的链接，怎么能保证它们的链接都正确，如果调整了图片或网页的存储路径，那不是全乱了吗？如何提高工作效率呢？



Dreamweaver 工具的站点管理功能，不但可以将绝对路径自动地转化为相对路径，并且当在站点中改动文件路径时，与这些文件关联的链接路径都会自动更改。

5.4.3　在网页中插入图像

图像可以美化网页，插入图像使用单标记<img>。img 标记的属性及描述如下表所示。

属性	值	描述
alt	text	定义有关图形的描述
src	URL	要显示的图像的 URL
height	pixels %	定义图像的高度
ismap	URL	把图像定义为服务器端的图像映射
usemap	URL	定义作为客户端图像映射的一幅图像；请参阅<map>和<area>标签，了解其工作原理
vspace	pixels	定义图像顶部和底部的空白，请使用 CSS 代替
width	pixels %	设置图像的宽度

1．插入图像

src 属性用于指定图片源文件的路径，它是 img 标记必不可少的属性。语法格式如下：

```
<img src="图片路径">
```

图片的路径可以是绝对路径，也可以是相对路径。下面的实例是在网页中插入图片。

【例 5.13】（实例文件：ch05\5.13.html）

```
 <!DOCTYPE html>
<html>
<head>
<title>插入图片</title>
</head>
 <body>
<img src="images/meishi.jpg">
</body>
</html>
```



图 5-17　插入图片

在 IE 9.0 中的预览效果如图 5-17 所示。

2．从不同位置插入图像

在插入图片时，用户可以将其他文件夹或服务器的图片显示到网页中。

【例 5.14】（实例文件：ch05\5.14.html）

```
<!DOCTYPE html>
<html>
<body>
<p>
```

```
来自一个文件夹的图像:
<img src="images/meishi.jpg" />
</p>
<p>
来自 baidu 的图像:
<img src="http://www.baidu.com/img/shouye_b5486898c692066bd2cbaeda86d74448.gif" />
</p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-18 所示。



图 5-18　插入图片

5.5　编辑图像

在插入图片时，用户还可以编辑网页中的图像。

5.5.1　设置图像的宽度和高度

在 HTML 文档中，还可以设置插入图片的大小，一般是按原始尺寸显示，也可以设置显示尺寸。设置图像尺寸分别用属性 width（宽度）和 height（高度）。

【例 5.15】（实例文件：ch05\5.15.html）

```
<!DOCTYPE html>
<html>
<head>
<title>插入图片</title>
</head>
<body>
<img src="images/meishi.jpg">
```

```
<img src="images/meishi.jpg" width="200">
<img src="images/meishi.jpg" width="200" height="300">
</body>
</html
```

在 IE 9.0 中的预览效果，如图 5-19 所示。

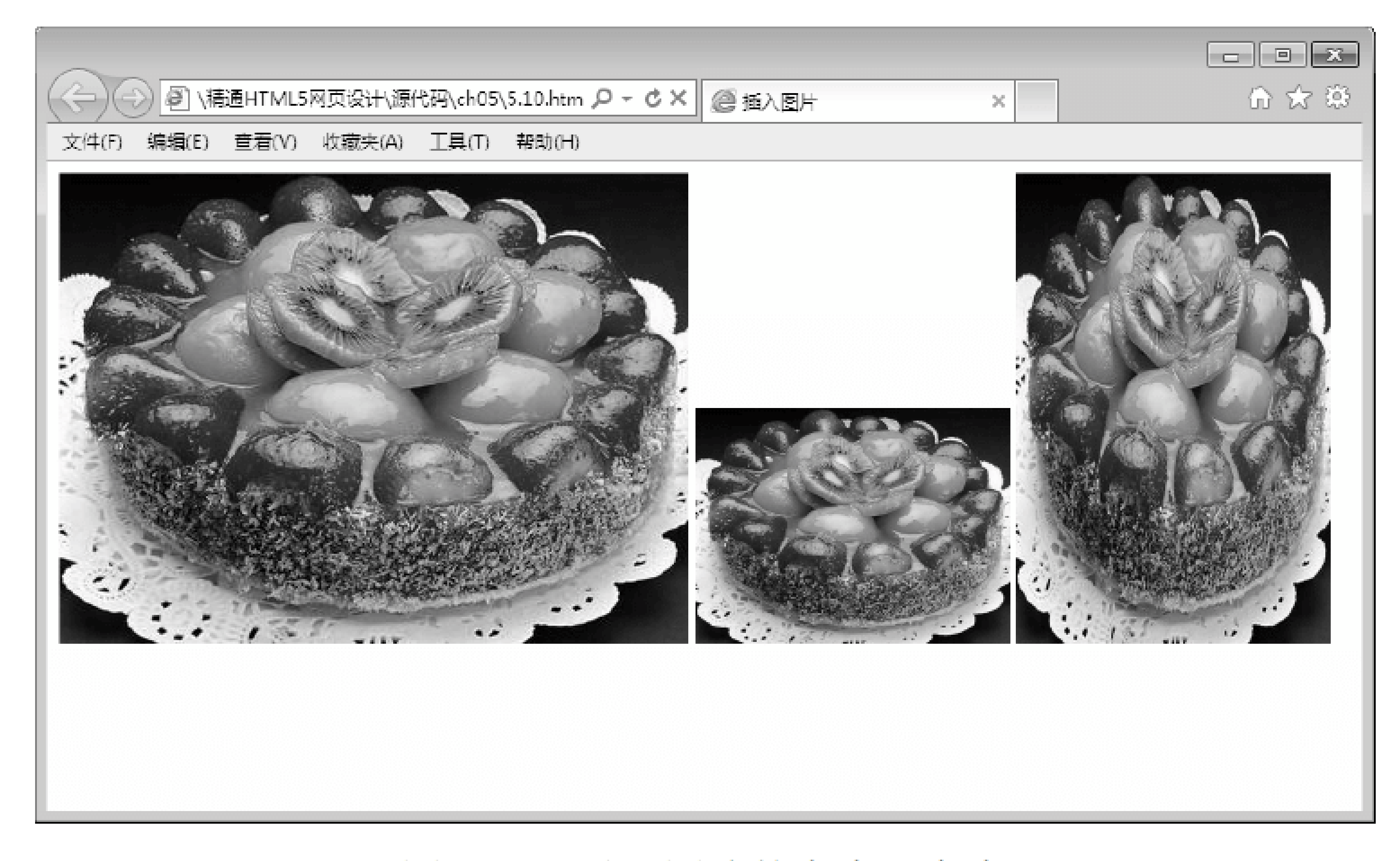


图 5-19 设置图片的宽度和高度

由图可以看到，图片的显示尺寸是由 width（宽度）和 height（高度）控制的。当只为图片设置一个尺寸属性时，另外一个尺寸就以图片原始的长宽比例来显示。图片的尺寸单位可以选择百分比或数值。百分比为相对尺寸，数值是绝对尺寸。



注 意

对于网页中插入的图像都是位图，放大尺寸，图像会出现马赛克，变得模糊。



提 示

在 Windows 中查看图片的尺寸，只需要找到图像文件，把鼠标指针移动到图像上，停留几秒后，就会出现一个提示框，说明图像文件的尺寸。尺寸后显示的数字，代表图像的宽度和高度，如 256 × 256。

5.5.2 设置图像的提示文字

图像提示文字的作用有两个：一是当浏览网页时，如果图像下载完成，将鼠标指针放在该图像上，鼠标指针旁边会出现提示文字，为图像添加说明性文字；二是，如果图像没有成功下载，在图像的位置上就会显示提示文字。

随着互联网技术的发展，网速已经不是制约因素，因此一般都能成功下载图像。现在，alt 还有另外一个作用，即在百度、Google 等大搜索引擎中，搜索图片不如文字方便，如果给图片添加适当提示，可以方便搜索引擎的检索。

下面实例将为图片添加提示文字效果。

【例 5.16】（实例文件：ch05\5.16.html）

```
<!DOCTYPE html>
<html>
<head>
<title>图片文字提示</title>
</head>
<body>
<img src="images/meishi.jpg" alt="草莓甜橙沙拉">
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-20 所示。用户将鼠标放在图片上时，即可看到提示文字。



图 5-20　图片文字提示



在 Firefox 中不支持该功能。

5.5.3　设置图片为网页背景

在插入图片时，用户可以根据需要将某些图片设置为网页的背景。GIF 和 JPG 文件均可用作 HTML 背景。如果图像小于页面，图像会进行排列。

【例 5.17】（实例文件：ch05\5.17.html）

```
<!DOCTYPE html>
<html>
<body background="images/background.jpg">
<h3>图像背景</h3>
```

```
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-21 所示。



图 5-21　图片背景

5.5.4　排列图像

在网页的文字当中，如果插入图片，这时可以对图像进行排序。常用的排序方式为居中、底部对齐、顶部对齐。

【例 5.18】（实例文件：ch05\5.18.html）

```
<!DOCTYPE html>
<html>
<body>
<h2>未设置对齐方式的图像：</h2>
<p>图像<img src ="images/logo.gif">在文本中</p>
<h2>已设置对齐方式的图像：</h2>
<p>图像<img src=" images/logo.gif " align="bottom">在文本中</p>
<p>图像<img src =" images/logo.gif " align="middle">在文本中</p>
<p>图像<img src =" images/logo.gif " align="top">在文本中</p>
</body>
</html>
```

在 IE 9.0 中的预览效果如图 5-22 所示。

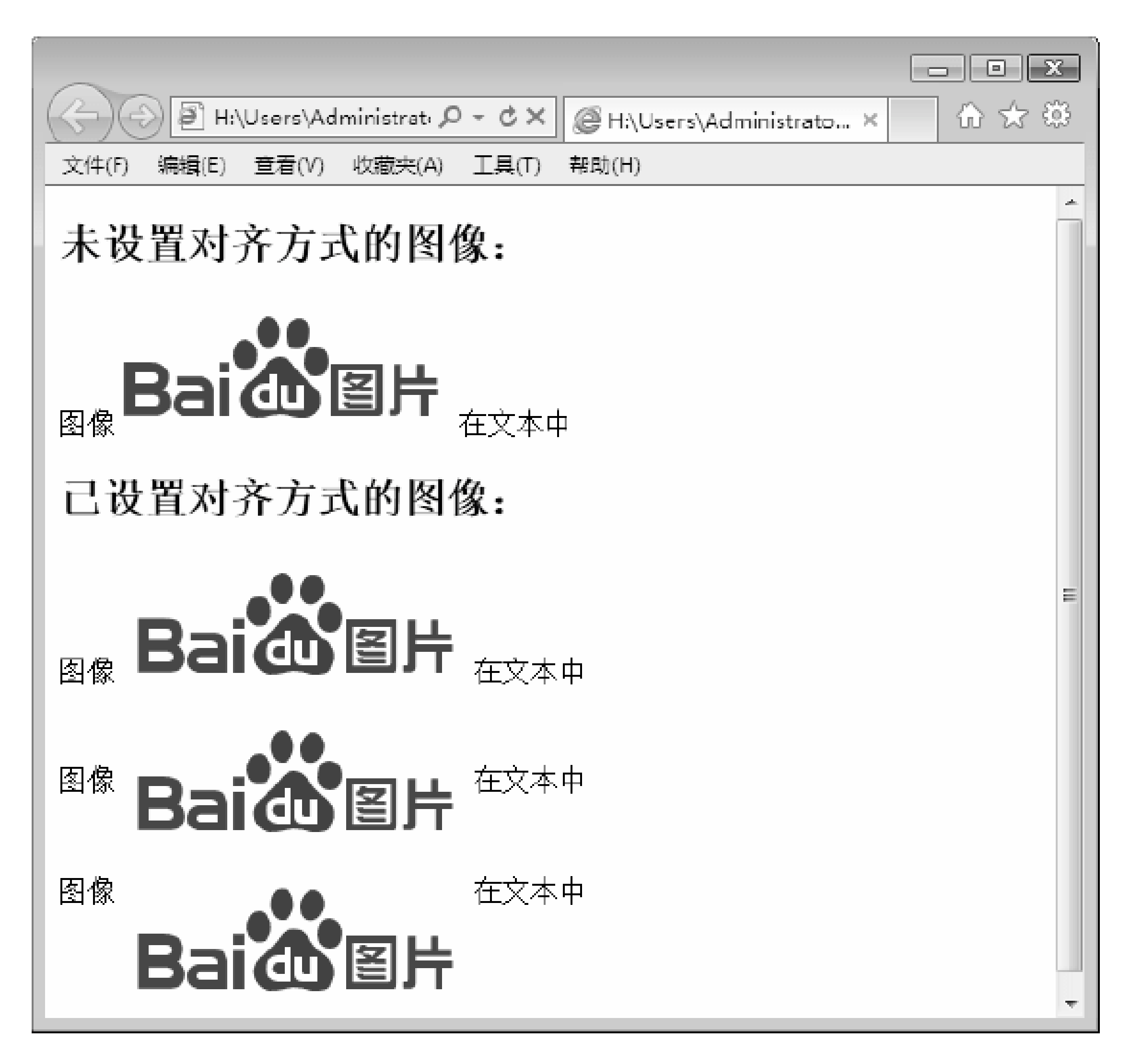


图 5-22　图片对齐方式



bottom 对齐方式是默认的对齐方式。

5.6　综合实例——图文并茂的房屋装饰装修网页

本章讲述了网页组成元素中最常用的文本和图片。本综合实例要创建一个由文本和图片构成的房屋装饰效果网页，如图 5-23 所示。



图 5-23　房屋装饰效果网页

具体操作步骤如下：

01 在 Dreamweaver CS5.5 中新建 HTML 文档，并修改成 HTML 5 标准，代码如下：

```
<!DOCTYPE html>
<html >
<head>
<title>房屋装饰装修效果图</title>
</head>
<body>
</body>
</html>
```

02 在 body 部分增加如下 HTML 代码保存页面：

```
<p><img src="images/xiyatu.jpg" width="300" height="200"/><img src="images/stadshem.jpg" width="300" height="200"/><br />
西雅图原生态公寓室内设计 与 Stadshem 小户型公寓设计（带阁楼）</p>
<hr/>
<p><img src="images/qingxinhuoli.jpg" width="300" height="200"/><img src="images/renwen.jpg" width="300" height="200"/><br />
清新活力家居与人文简约悠然家居</p>
<hr />
```



<hr>标记的作用是定义内容中的主题变化，并显示为一条水平线，在 HTML 5 中它没有任何属性。

另外，快速插入图片及设置相关属性，可以借助 Dreamweaver CS5.5 的插入功能，或按下快捷键 Ctrl+Alt+I。

5.7 问题解答

1. 换行标记和段落标记有哪些区别？

换行标记是单标记，不能写结束标记。段落标记是双标记，可以省略结束标记，也可以不省略。默认情况下，段落之间的距离和段落内部的行间距是不同的，段落间距比较大，行间距比较小。HTML 无法调整段落间距和行间距，如果希望调整它们，就必须使用 CSS。在 Dreamweaver CS5.5 的设计视图下，按下 Enter 键可以快速换段，按下 Shift+ Enter 键可以快速换行。

2. 无序列表<ul>元素具有什么作用？

无序列表元素主要用于条理化和结构化文本信息。在实际开发中，无序列表在制作导航菜单时使用广泛，导航菜单的结构一般都使用无序列表实现。

3．在浏览器中，图片无法显示，为什么？

图片在网页中属于嵌入对象，并不是保存在网页中的，网页只是保存了指向图片的路径。浏览器在解释 HTML 文件时，会按指定的路径去寻找图片，如果在指定的位置不存在图片，就无法正常显示。为了保证图片的正常显示，制作网页时需要注意以下几处。

（1）图片格式一定是网页支持的；

（2）图片的路径一定要正常，并且图片文件扩展名不能省略；

（3）HTML 文件位置发生改变时，图片一定要随着改变，即图片位置和 HTML 文件位置始终保持相对一致。

第 6 章　使用 HTML 5 建立超链接

HTML 文件中最重要的应用之一就是超链接，超链接是一个网站的灵魂，Web 上的网页是互相链接的，单击超链接的文本或图形就可以链接到其他页面。

6.1　URL

URL 就是人们通常说的“网址”，用于指定 Internet 上的资源位置。

6.1.1　URL 的格式

网络中的计算机之间是通过 IP 地址区分的，如果希望访问网络中某台计算机中的资源，首先要定位到这台计算机。IP 地址是由 32 位的（二进制），即 32 个 0/1 代码组成，数字之间没有意义，不容易记忆。为了方便记忆，现在计算机一般采用域名的方式来寻址，即在网络上使用一组具有特殊意义的字符组成的地址代替 IP 地址来访问网络资源。

URL 由 4 个部分组成，即“协议”、“主机名”、“文件夹名”和“文件名”，如图 6-1 所示。



图 6-1　URL 组成

互联网中有各种各样的应用，如 Web 服务、FTP 服务等。每种服务应用都对应相应的协议，通常通过浏览器浏览网页的协议都是 HTTP 协议，即“超文本传输协议”，因此通常网页的地址都以“http://”开头。

“www.baidu.com”为主机名，表示文件存在于哪台服务器 ，主机名可以通过 IP 地址或者域名来表示。

确定到主机后，还需要说明文件存在于这台服务器的哪个文件夹中，这里文件夹可以分为多个层级。

确定文件夹后，就要定位到文件，即要显示哪个文件，网页文件通常是以“.html”或“.htm”为扩展名。

6.1.2 URL 的类型

在第 5 章讲解网页中使用图像时，已经介绍了“路径”的概念。对于超链接来说，路径的概念同样存在。

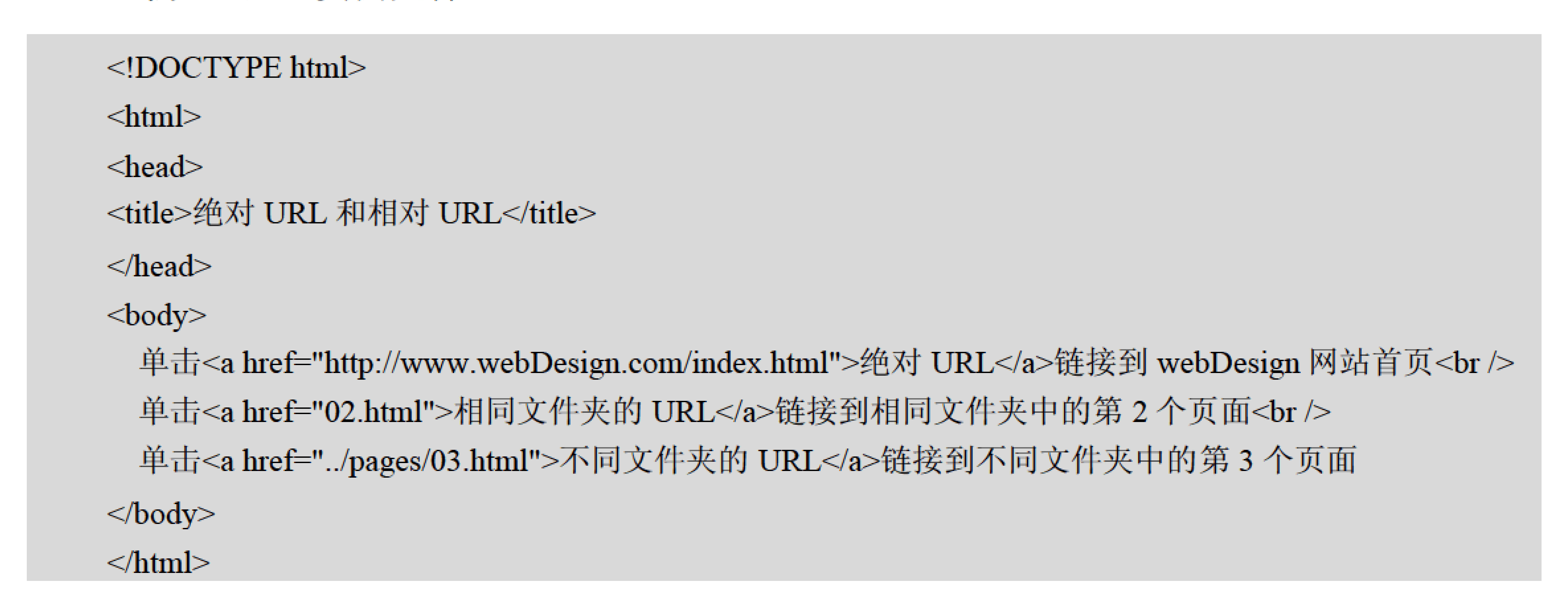
超链接的 URL 可以分为两种类型：“绝对 URL”和“相对 URL”。

（1）绝对 URL 一般用于访问非同一台服务器上的资源。

（2）相对 URL 是指访问同一台服务器上相同文件夹或不同文件夹中的资源。如果访问相同文件夹中的文件，只需要写文件名；如果访问不同文件夹中的资源，URL 以服务器的根目录为起点，指明文档的相对关系，由文件夹名和文件名两部分构成。

下面实例为使用绝对 URL 和相对 URL 实现超链接。

【例 6.1】（实例文件：ch06\6.1.html）



```
<!DOCTYPE html>
<html>
<head>
<title>绝对 URL 和相对 URL</title>
</head>
<body>
  单击<a href="http://www.webDesign.com/index.html">绝对 URL</a>链接到 webDesign 网站首页<br />
  单击<a href="02.html">相同文件夹的 URL</a>链接到相同文件夹中的第 2 个页面<br />
  单击<a href="../pages/03.html">不同文件夹的 URL</a>链接到不同文件夹中的第 3 个页面
</body>
</html>
```

在上述代码中，第一个链接使用的是绝对 URL；第二个使用的是服务器相对 URL，也就是链接到文档所在服务器的根目录下的 02.html；第三个使用的是文档相对 URL，即原文档所在文件夹的父文件夹下面的 pages 文件夹中的 03.html 文件。

在 IE 9.0 中预览网页效果如图 6-2 所示。

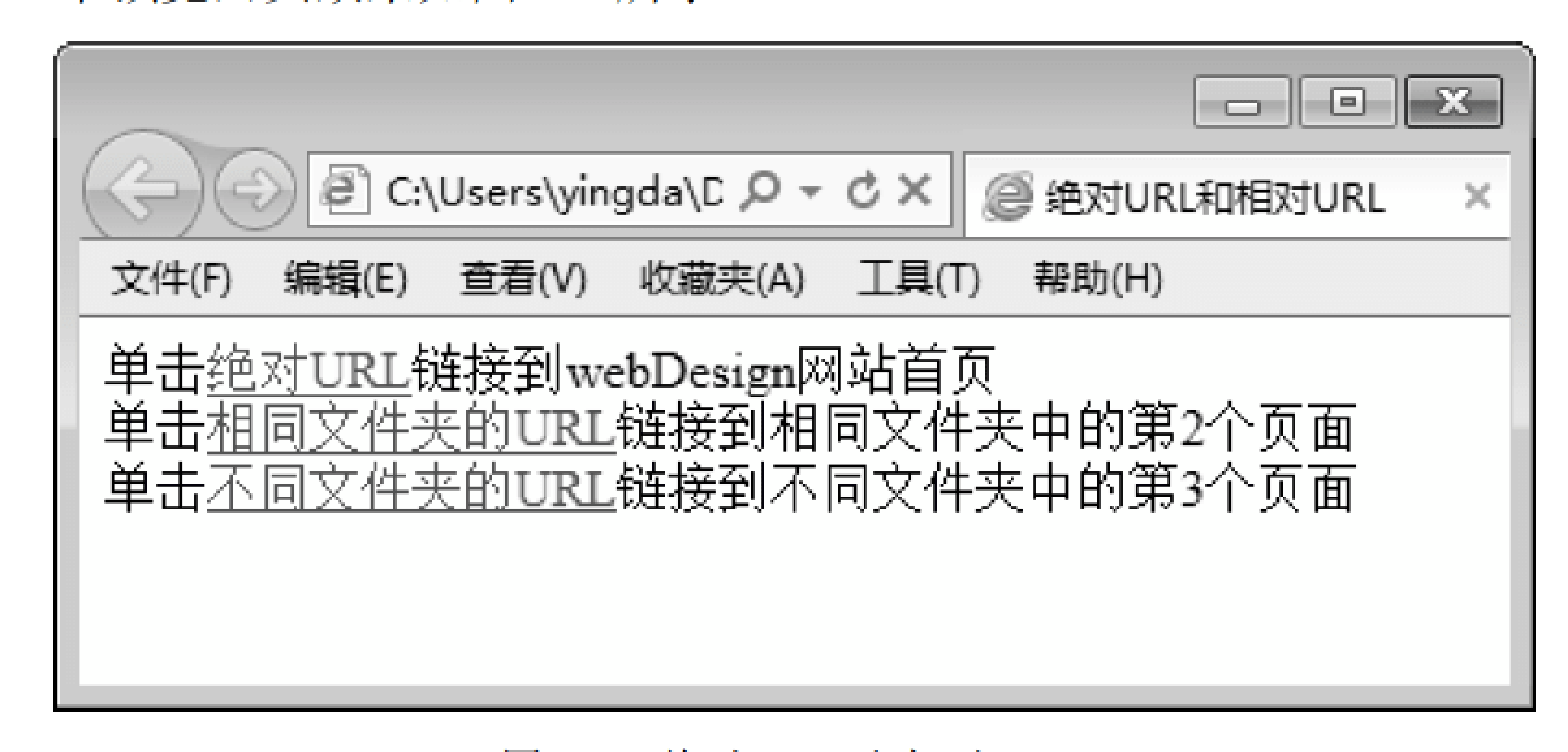


图 6-2　绝对 URL 和相对 URL

6.2　创建超链接

超链接就是当用鼠标单击一些文字、图片或其他网页元素时，浏览器就会根据其指示载入一个新的页面或跳转到页面的其他位置。超链接除了可链接文本外，也可链接各种媒体，如声音、图像、动画，通过它们可享受丰富多彩的多媒体世界。

建立超链接所使用的 HTML 标记为<a>和。超链接最重要的有两个要素：设置为超链接的网页元素和超链接指向的目标地址。超链接的基本结构如下：

```
<a href=URL>网页元素</a>
```

6.2.1　设置文本和图片的超链接

设置超链接的网页元素通常使用文本和图片。文本超链接和图片超链接通过<a>标记实现，将文本或图片放在<a>开始标记和结束标记之间即可建立超链接。下面实例将实现文本和图片的超链接。

【例 6.2】（实例文件：ch06\6.2.html）

```
<!DOCTYPE html>
<html>
<head>
<title>文本和图片超链接</title>
</head>
<body>
<a href="a.html"><img src="images/0371.gif"></a>
<a href="b.html">公司简介</a>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-3 所示，单击图片即可实现链接跳转的效果。



图 6-3　文本和图片链接效果



注 意

链接图片的外观在不同浏览器中的效果也不尽相同，例如在 Firefox 预览效果如图 6-4 所示。



图 6-4　Firefox 浏览器预览效果



注 意

在 Firefox 中，图片超链接不会增加边框，而在 IE 中，图片超链接会增加边框。

默认情况下，为文本添加超链接，文本会自动增加下划线，并且文本颜色变为蓝色，单击过的超链接，文本会变成暗红色。图片增加超链接以后，浏览器会自动给图片增加一个粗边框。

6.2.2　设置超链接指向的目标类型

通过上面的讲解，读者会发现超链接的目标对象都是“.html”类型的文件。超链接不但可以链接到各种类型（如图片文件、声音文件、视频文件、Word 文件等）的文件，还可以链接到其他网站、ftp 服务器、电子邮件等。

1．链接到各种类型的文件

超链接<a>标记的 href 属性是指向链接的目标，目标可以是各种类型的文件。如果是浏览器能够识别的类型，会直接在浏览器中显示；如果是浏览器不能识别的类型，在 IE 浏览器中会弹出文件下载对话框，如图 6-5 所示。



图 6-5　IE 中的文件下载对话框

【例 6.3】（实例文件：ch06\6.3.html）

```
<!DOCTYPE html>
<html>
<head>
<title>链接各种类型文件</title>
</head>
<body>
<p><a href="a.html">链接 html 文件</a></p>
<p><a href="coffe.jpg">链接图片</a></p>
<p><a href="2.doc">链接 Word 文档</a></p>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-6 所示。实现链接到 html 文件、图片和 Word 文档。



图 6-6　各种类型的链接

2. 链接到其他网站或 ftp 服务器

在网页中，友情链接也是推广网站的一种方式。下列代码实现了链接到其他网站或 ftp 服务器的目的。

```
<a href="http://www.baidu.com">链接百度</a>
<a href="ftp://172.16.1.254">链接到 ftp 服务器</a>
```



这里 ftp 服务器用的是 IP 地址。为了保证代码的正确运行，请读者填写有效的 ftp 服务器地址。

3. 设置电子邮件链接

在某些网页中，当访问者单击某个链接以后，会自动打开电子邮件客户端软件，如 Outlook 或 Foxmail 等，向某个特定的 E-mail 地址发送邮件，这个链接就是电子邮件链接。电子邮件链接的格式如下：

```
<a href="mailto:电子邮件地址" >网页元素</a>
```

【例 6.4】（实例文件：ch06\6.4.html）

```
<!DOCTYPE html>
<html>
<head>
<title>电子邮件链接</title>
</head>
<body>
<img src="images/logo.gif" width="119" height="49">  [免费注册][登录]
<a href="mailto:kfdzsj@126.com">站长信箱</a>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-7 所示，实现了电子邮件链接。



图 6-7 链接到电子邮件

当读者单击“站长信箱”链接时，会自动弹出 Outlook 窗口，要求编写电子邮件，如图 6-8 所示。

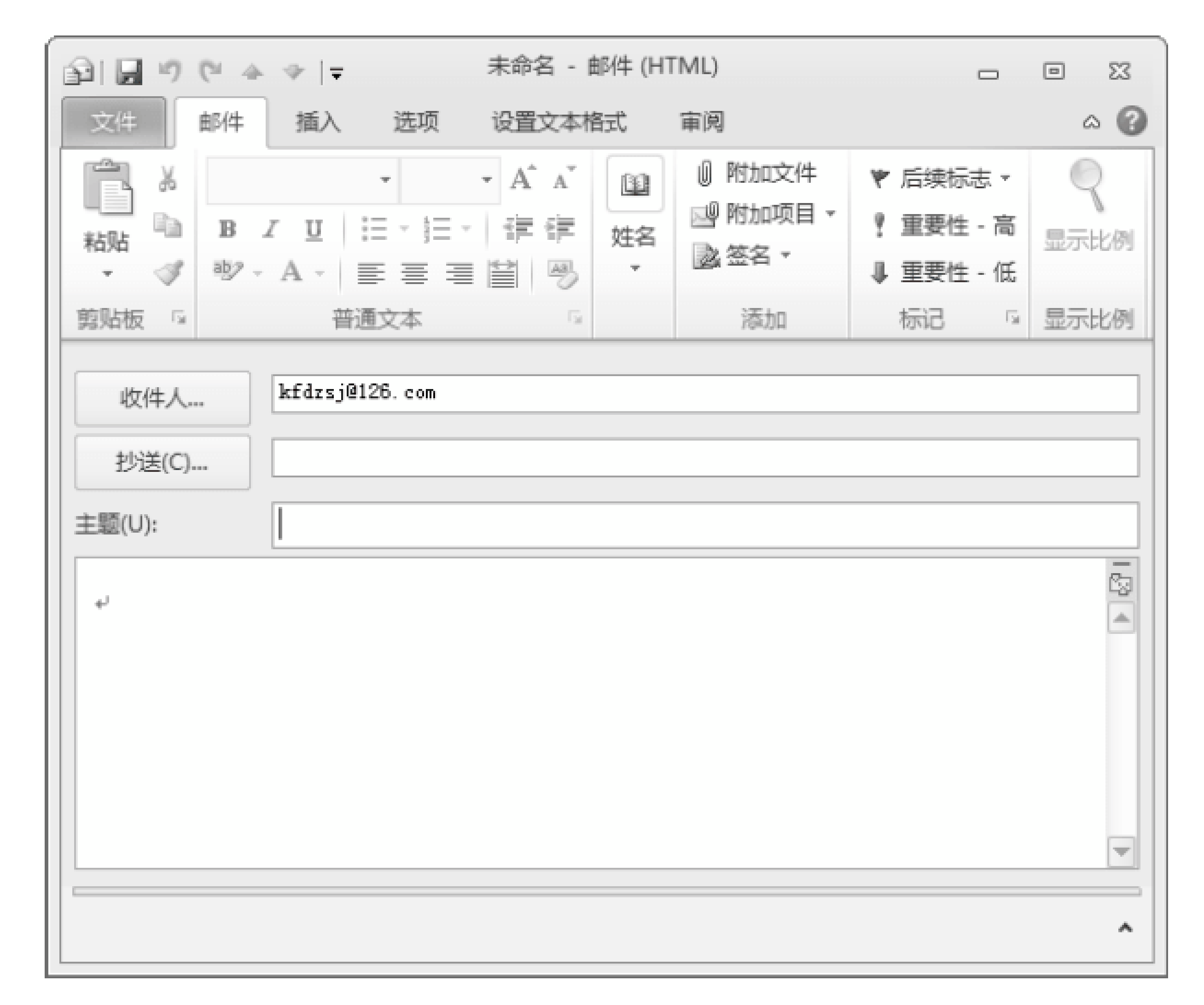


图 6-8　Outlook 新邮件窗口

6.2.3　用新窗口显示超链接页面

默认情况下，当单击超链接时，目标页面会在当前窗口中显示，替换当前页面的内容。如果要在单击某个链接以后，打开新的浏览器窗口并在这个新窗口中显示目标页面，就需要使用<a>标签的 target 属性。

target 属性的取值有 4 个，分别是“_blank”、“_self”、“_top”和“_parent”。由于 HTML 5 不再支持框架，所以“_top”、“_parent”这两个取值不常用。本节仅为读者讲解“_blank”、“_self”值。其中，“_blank”值为在新窗口中显示超链接页面；“_self”代表在自身窗口中显示超链接页面，当省略 target 属性时，默认取值为“_self”。

【例 6.5】（实例文件：ch06\6.5.html）

```
<!DOCTYPE html>
<html>
<head>
<title>以新窗口方式打开</title>
</head>
<body>
<a href="a.html target="_blank">新窗口</a>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-9 所示。



图 6-9 新窗口页面

6.2.4 如何链接到同一页面的不同位置

在建立链接时，尤其是文字比较多的网页，需要在同一页面的不同位置进行链接，这时就需要建立同一网页内的链接。

【例 6.6】（实例文件：ch06\6.6.html）

```
<!DOCTYPE html>
<html>
<body>
<p>
<a href="#C4">查看 第 4 章。</a>
</p>
<h2>第 1 章</h2>
<p>本章讲解图片相关知识……</p>
<h2>第 2 章</h2>
<p>本章讲解文字相关知识……</p>
<h2>第 3 章</h2>
<p>本章讲解动画相关知识……</p>
<h2><a name="C4">第 4 章</a></h2>
<p>本章讲解图形相关知识……</p>
<h2>第 5 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 6 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 7 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 8 章</h2>
<p>本章讲解列表相关知识……</p>
```

```
<h2>第 9 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 10 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 11 章</h2>
<p>本章讲解列表相关知识……</p>
<h2>第 12 章</h2>
<p>本章讲解列表相关知识……</p>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-10 所示。

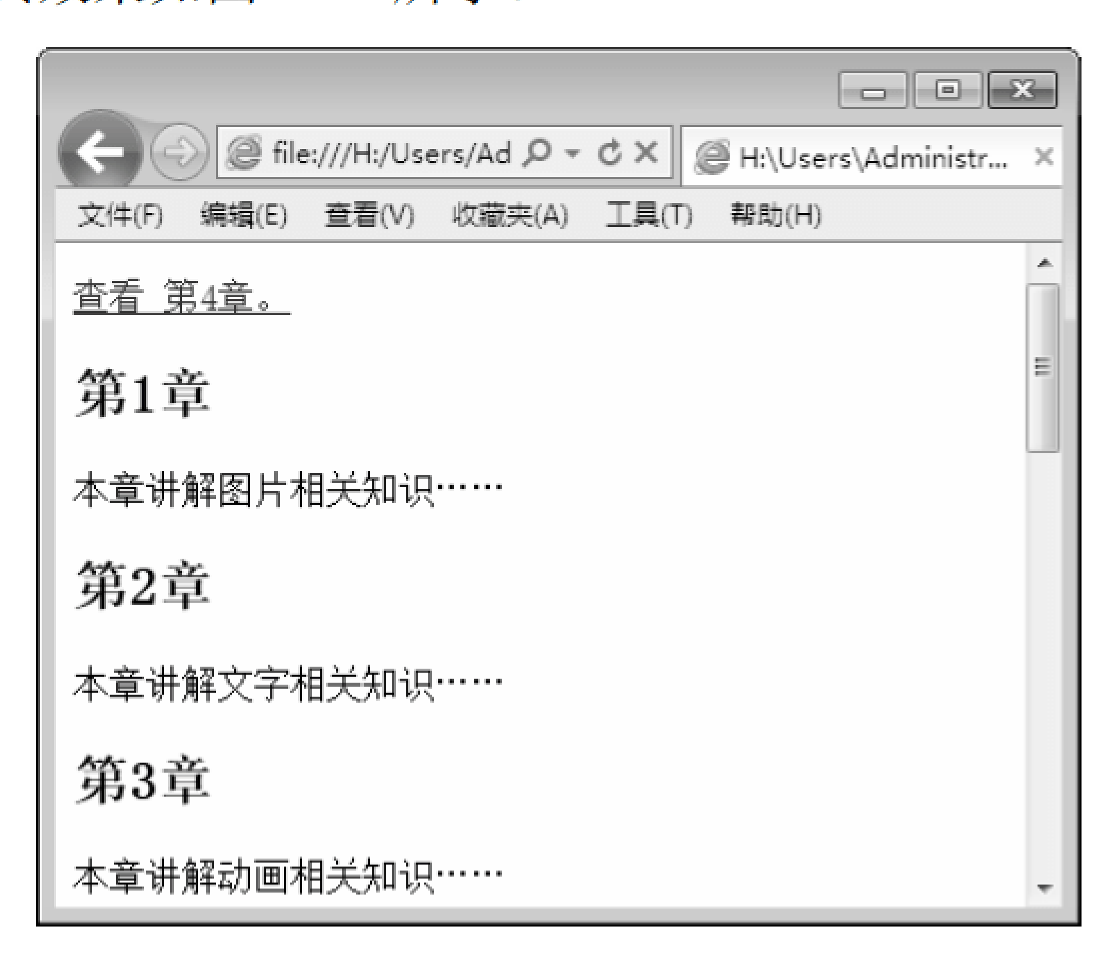


图 6-10　链接页面

单击页面中的链接，即可将“第 4 章”的内容跳转到页面顶部，如图 6-11 所示。



图 6-11　链接到“第 4 章”

6.3 创建热点区域

在浏览网页时，读者会发现，当单击一张图片的不同区域，会显示不同的链接内容，这就是图片的热点区域。所谓图片的热点区域就是将一个图片划分成若干个链接区域。访问者单击不同的区域会链接到不同的目标页面。

在 HTML 中，可以为图片创建三种类型的热点区域：矩形、圆形和多边形。创建热点区域使用标记<map>和<area>，语法格式如下：

```
<img src="图片地址" usemap="#名称">
<map id="#名称">
  <area shape="rect" coords="10,10,100,100" href="#">
  <area shpe="circle" cords="120,120,50" href="#">
  <area shape="poly" coords="78,13,81,14,53,32,86,38" href="#">
</map>
```

在上面的语法格式中，需要读者注意以下几点。

（1）要想建立图片热点区域，必须先插入图片。注意，图片必须增加 usemap 属性，说明该图像是热点区域映射图像，属性值必须以“#”开头，加上名字，如“#pic”。那么上面一行代码可以修改为：<img src="图片地址" usemap="#pic">。

（2）<map>标记只有一个属性 id，其作用是为区域命名，其设置值必须与<img>标记的 usemap 属性值相同，修改上述代码为：<map id="#pic">

（3）<area>标记主要是定义热点区域的形状及超链接，它有三个必须的属性。

- shape 属性，控件划分区域的形状，其取值有三个，分别是 rect（矩形）、circle（圆形）和 poly（多边形）。
- coords 属性，控制区域的划分坐标。
  - 如果 shape 属性取值为 rect，那么 coords 的设置值分别为矩形在左上角 x、y 坐标点和右下角 x、y 坐标点，单位为像素。
  - 如果 shape 属性取值为 cirle，那么 coords 的设置值分别为圆形的圆心 x、y 坐标点和半径值，单位为像素。
  - 如果 shape 属性取值为 poly，那么 coords 的设置值分别为多边形在各个点的 x、y 坐标，单位为像素。
- href 属性是为区域设置超链接的目标，设置值为“#”时，表示为空链接。

6.4 创建浮动框架

HTML 5 中已经不支持 frameset 框架，但是它仍然支持 iframe 浮动框架的使用。浮动框架可以自由控制窗口大小，可以配合表格随意地在网页中的任何位置插入窗口，实际上就是在窗口中再创建一个窗口。

使用 iframe 创建浮动框架的格式如下：

```
<iframe src="链接对象" >
```

其中，src 表示浮动框架中显示对象的路径，可以是绝对路径，也可以是相对路径。例如，下面的代码是在浮动框架中显示百度网站。

【例 6.7】（实例文件：ch06\6.7.html）

```
<!DOCTYPE html>
<html>
<head>
<title>浮动框架中显示百度网站</title>
</head>
<body>
<iframe src="http://www.baidu.com"></iframe>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 6-12 所示。



图 6-12　浮动框架效果

从预览结果可见，浮动框架在页面中又创建了一个窗口，默认情况下，浮动框架的宽度和高度为 220×120 像素。如果需要调整浮动框架尺寸，需要使用 CSS 样式；若修改上述浮动框架尺寸，要在 head 标记部分增加如下 CSS 代码：

```
<style>
iframe{
    width:600px;     //宽度
    height:800px;   //高度
    border:none;    //无边框
}
</style>
```

在 IE 9.0 中预览网页效果如图 6-13 所示。



图 6-13　修改宽度和高度的浮动框架



在 HTML 5 中，iframe 仅支持 src 属性，再无其他属性。

6.5　综合实例——用 Dreamweaver 精确定位热点区域

上面讲述了 HTML 创建热点区域的方法，但是最让读者头痛的地方，就是坐标点的定位。对于简单的形状还可以，如果形状较多且形状复杂，确定坐标点这项工作的工程量就很大，因此，不建议使用 HTML 代码去完成。这里将为读者介绍一个快速且能精确定位热点区域的方法。在 Dreamweaver CS5.5 可以很方便的实现这个功能。

使用 Dreamweaver CS5.5 创建图片热点区域的具体操作步骤如下：

01 创建一个 HTML 文档，插入一张图片文件，如图 6-14 所示。



图 6-14　插入图片

02 选择图片，在 Dreamweaver CS5.5 中打开“属性”面板，面板左下角有三个图标按钮，依次代表矩形、圆形和多边形热点区域。单击左边的“矩形热点”工具图标，如图 6-15 所示。



图 6-15　Dreamweaver CS5.5 中图像的“属性”面板

03 将鼠标指针移动到被选中的图片，以“创意信息平台”栏中的矩形大小为准，按下鼠标左键，从左上方向右下方拖曳鼠标，得到矩形区域，如图 6-16 所示。



图 6-16　绘制矩形热点区域

04 绘制出来的热点区域呈现出半透明状态，效果如图 6-17 所示。

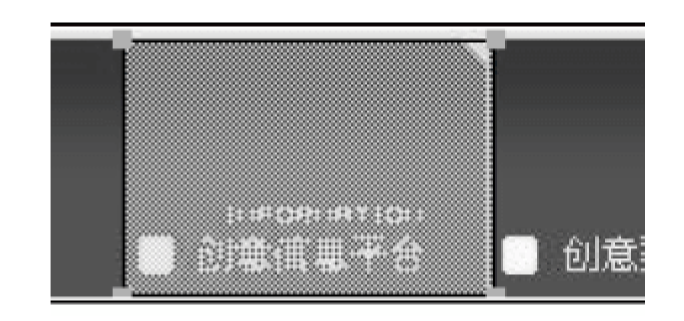


图 6-17　完成矩形热点区域的绘制

05 如果绘制出来的矩形热点区域有误差，可以通过“属性”面板中的“指针热点”工具进行编辑，如图 6-18 所示。



图 6-18　“指针热点”工具

06 完成上述操作之后，保持矩形热点区域被选中状态，然后在“属性”面板中的“链接”文本框中输入该热点区域链接对应的跳转目标页面。

07 在“目标”下拉列表框中有 4 个选项，它们决定着链接页面的弹出方式，这里如果选择了“_blank”，那么矩形热点区域的链接页面将在新的窗口中弹出。如果“目标”选项保持空白，就表示仍在原来的浏览器窗口中显示链接的目标页面，这样，矩形热点区域就设置好了。

08 接下来继续为其他菜单项创建矩形热点区域。操作方法请参阅步骤 2~步骤 7，完成后的效果如图 6-19 所示。

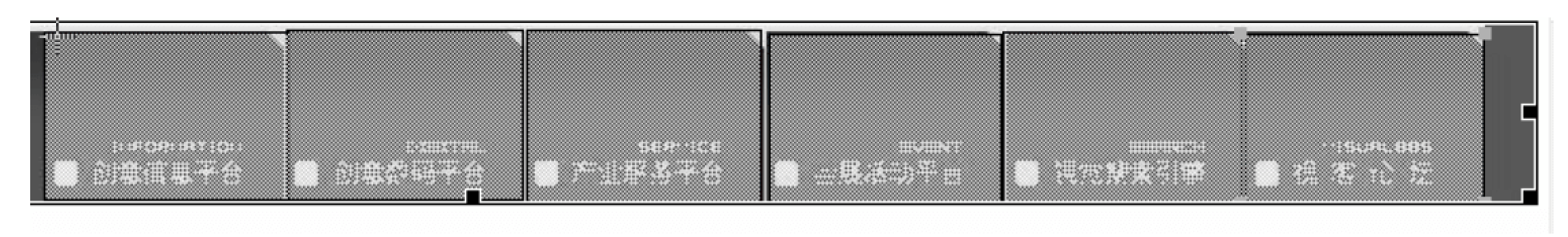


图 6-19　为其他菜单项创建矩形热点区域

09 完成后保存并预览页面。可以发现，凡是绘制了热点的区域，鼠标指针移上去时就会变成手形，单击就会跳转到相应的页面。

10 到此，网站的导航就使用热点区域制作完成了。查看此时页面相应的 HTML 源代码

如下：

```
<!DOCTYPE html>
<html>
<head>
<title>创建热点区域</title>
</head>
<body>
<img src="images/04.jpg" width="1001" height="87" border="0" usemap="#Map">
<map name="Map">
  <area shape="rect" coords="298,5,414,85" href="#">
  <area shape="rect" coords="412,4,524,85" href="#">
  <area shape="rect" coords="525,4,636,88" href="#">
  <area shape="rect" coords="639,6,749,86" href="#">
  <area shape="rect" coords="749,5,864,88" href="#">
  <area shape="rect" coords="861,6,976,86" href="#">
</map>
</body>
</html>
```

可以看到，Dreamweaver CS5.5 自动生成的 HTML 代码结构和前面介绍的是一样的，但是所有的坐标都自动计算出来了，这正是网页制作工具的快捷之处。使用这些工具本质上和手工编写 HTML 代码没有区别，只是使用这些工具可以提高工作效率。



注 意

本书所讲述的手工编写 HTML 代码，在 Dreamweaver CS5.5 工具中几乎都有对应的操作，请读者自行研究，以提高编写 HTML 代码效率。但是，请读者注意，使用网页制作工具前，一定要明白这些 HTML 标记的作用。因为一个专业的网页设计师必须具备 HTML 方面的知识，不然再强大的工具也只能是无根之树，无源之泉。

参照矩形热区的操作方法，为图 6-20 创建圆形和多边形热点区域。

6.6 问题解答

1．在创建超链接时，使用绝对 URL 还是相对 URL？

在创建超链接时，如果要链接的是另外一个网站中的资源，需要使用完整的绝对 URL；如果在网页中创建内部链接，一般使用相对当前文档或站点根文件夹的相对 URL。

2．链接增多后的网站，如何设置目录结构以方便维护？

当一个网站的网页数量增加到一定程度以后，网站的管理与维护将变得非常繁琐，因此掌握一些网站管理与维护的技术是非常实用的，可以节省很多时间。建立适合的网站文件存储结

构，可以方便网站的管理与维护。通常使用的三种网站文件组织结构方案及文件管理遵循的原则如下：

- 按照文件的类型进行分类管理。将不同类型的文件放在不同的文件夹中，这种存储方法适合于中小型的网站，这种方法是通过文件的类型对文件进行管理的。
- 按照主题对文件进行分类。网站的页面按照不同的主题进行分类存储。同一主题的所有文件存放在一个文件夹中，然后再进一步细分文件的类型，这种方案适用于页面与文件数量众多、信息量大的静态网站。
- 对文件类型进行进一步细分存储管理。这种方案是第一种存储方案的深化，将页面进一步细分后进行分类存储管理。这种方案适用于文件类型复杂、包含各种文件的多媒体动态网站。

第 7 章　使用 HTML 5 创建表格

在 HTML 中表格不但可以清晰地显示数据，而且可以用于页面布局。HTML 中表格类似于 Word 软件中的表格，尤其是使用网页制作工具，操作很相似。HTML 制作表格的原理是使用相关标记，如表格对象 table 标记、行对象 tr、单元格对象 td 才能完成。

7.1　表格的基本结构

使用表格显示数据，可以更直观和更清晰。在 HTML 文档中表格主要用于显示数据，虽然可以使用表格布局，但是不建议使用，它有很多弊端。表格一般由行、列和单元格组成，如图 7-1 所示。

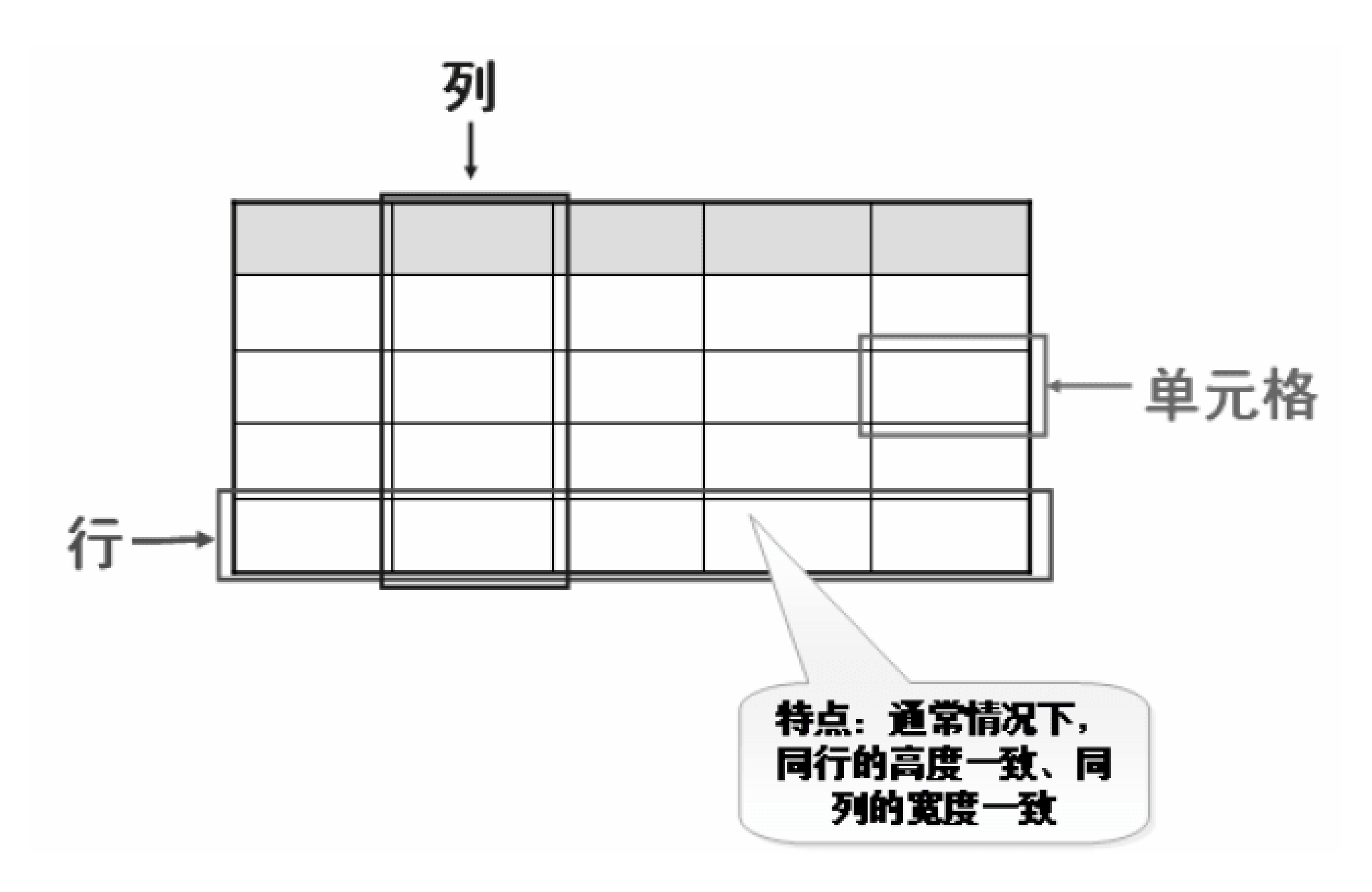


图 7-1　表格的组成

在 HTML 5 中用于标记表格的标记如下：

- <table>标记：用于标识一个表格对象的开始，</table>标记标识一个表格对象的结束。一个表格中，只允许出现一对<table>标记。在 HTML 5 中不再支持它的任何属性。
- <tr>标记：用于标识表格一行的开始，</tr>标记用于标识表格一行的结束。表格内有多少对<tr></tr>标记，就表示表格中有多少行。在 HTML 5 中不再支持它的任何属性。
- <td>标记：用于标识表格某行中的一个单元格开始，</td>标记用于标识表格某行中的一个单元格结束。<td></td>标记书写在<tr></tr>标记内，一对<tr></tr>标记内有多少对<td></td>标记，就表示该行有多少个单元格。在 HTML 5 中仅有 colspan 和 rowspan 两个属性。

最基本的表格，必须包含一对<table></table>标记、一对或几对<tr></tr>标记以及一对或几对<td></td>标记。一对<table></table>标记定义一个表格，一对<tr></tr>标记定义一行，一对<td></td>标记定义一个单元格，例如定义一个 4 行 3 列的表格。

【例 7.1】（实例文件：ch07\7.1.html）

```
<!DOCTYPE html>
<html>
<head>
<title>表格基本结构</title>
</head>
<body>
<table border="1">
 <tr>
    <td>A1</td>
    <td>B1</td>
    <td>C1</td>
 </tr>
 <tr>
    <td>A2</td>
    <td>B2</td>
    <td>C2</td>
 </tr>
 <tr>
    <td>A3</td>
    <td>B3</td>
    <td>C3</td>
 </tr>
 <tr>
    <td>A4</td>
    <td>B4</td>
    <td>C4</td>
 </tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 7-2 所示。



提 示

从预览图中，读者会发现，表格没有边框，行高及列宽也无法控制。讲述上述知识时，提到 HTML 5 中除了 td 标记提供两个单元格合并属性之外，<table>和<tr>标记也没有任何属性。



图 7-2　表格基本结构

7.2　表格的基本操作

了解了表格的基本结构后，下面来介绍表格的基本操作，主要包括创建表格、设置表格的边框类型、设置表格的表头、合并单元等。

7.2.1　创建表格

表格可以分为普通表格以及带有标题的表格两种，可以在 HTML 5 中创建这两种表格。

1. 创建普通表格

例如创建 1 列、1 行 3 列和 2 行 3 列等三个表格。

【例 7.2】（实例文件：ch07\7.2.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>一列：</h4>
<table border="1">
<tr>
  <td>100</td>
</tr>
</table>
<h4>一行三列：</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
</table>
```

```
<h4>两行三列：</h4>
<table border="1">
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页效果如图 7-3 所示。



图 7-3　程序运行结果

2. 创建带有标题的表格

有时，为了方便表述表格，还需要在表格上面加上标题，例如创建一个带有标题的表格。

【例 7.3】（实例文件：ch07\7.3.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>带有标题的表格</h4>
<table border="3">
<caption>数据统计表</caption>
<tr>
```

```
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-4 所示。



图 7-4　程序运行结果

7.2.2　定义表格的边框类型

使用表格的 border 属性可以定义表格的边框类型，如常见的加粗边框表格，例如创建不同边框类型的表格。

【例 7.4】（实例文件：ch07\7.4.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>普通边框</h4>
<table border="1">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
```

```
  <td>Row</td>
</tr>
</table>
<h4>加粗边框</h4>
<table border="8">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-5 所示。



图 7-5　程序运行结果

7.2.3　定义表格的表头

表格当中也存在有表头，常见的表头分为垂直与水平两种，例如分别创建带有垂直和水平表头的表格。

【例 7.5】（实例文件：ch07\7.5.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>水平的表头</h4>
<table border="1">
<tr>
```

```
    <th>姓名</th>
    <th>性别</th>
    <th>电话</th>
  </tr>
  <tr>
    <td>张三</td>
    <td>男</td>
    <td>123456</td>
  </tr>
  </table>
  <h4>垂直的表头：</h4>
  <table border="1">
  <tr>
    <th>姓名</th>
    <td>小丽</td>
  </tr>
  <tr>
    <th>性别</th>
    <td>女</td>
  </tr>
  <tr>
    <th>电话</th>
    <td>123456</td>
  </tr>
  </table>
  </body>
  </html>
```

在 IE 9.0 中预览网页的效果如图 7-6 所示。

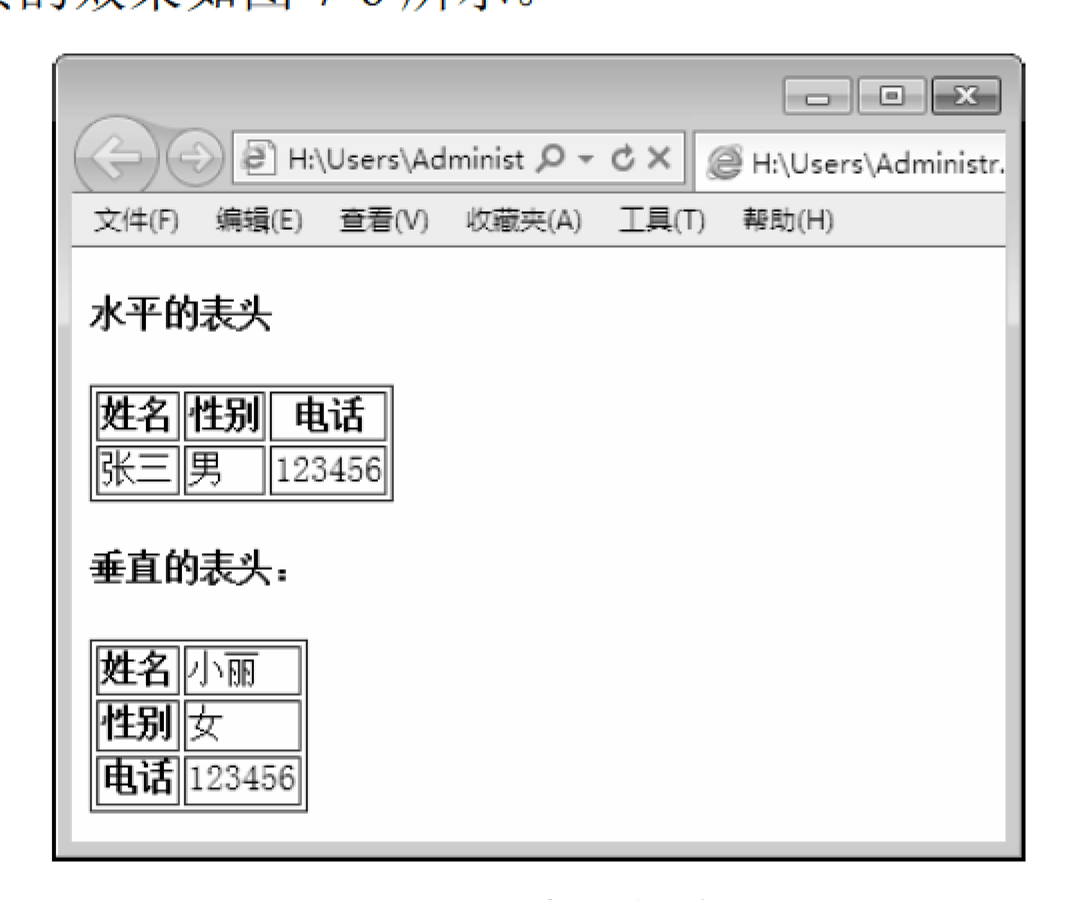


图 7-6 程序运行结果

7.2.4　设置表格背景

当创建好表格后，为了美观还可以设置表格的背景。

1. 定义表格背景颜色

为表格添加背景颜色是美化表格的一种方式，例如为表格添加背景颜色。

【例 7.6】（实例文件：ch07\7.6.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>背景颜色：</h4>
<table border="1"
bgcolor="green">
<tr>
  <td>100</td>
  <td>200</td>
</tr>
<tr>
  <td>300</td>
  <td>400</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-7 所示。

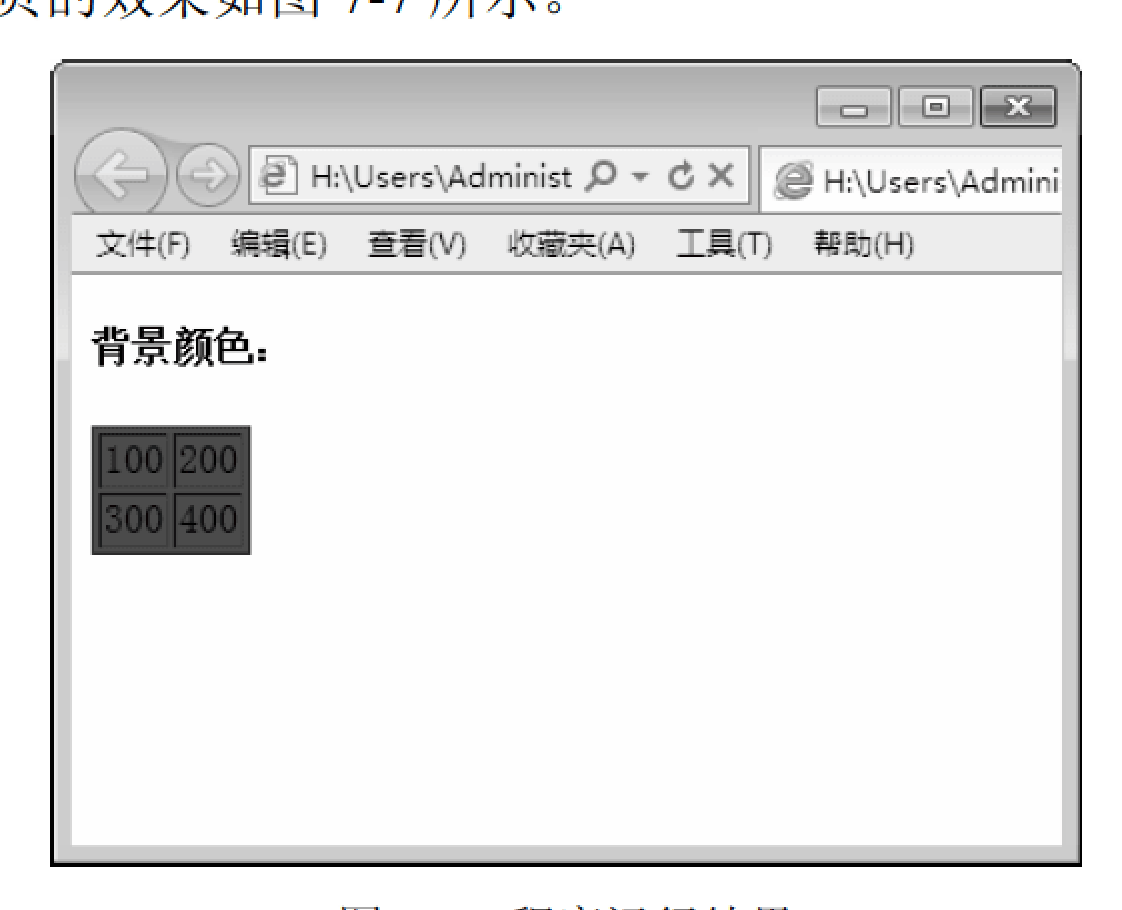


图 7-7　程序运行结果

2. 定义表格背景图片

除了可以为表格添加背景颜色外，还可以将图片设置为表格的背景，例如为表格添加背景

图片。

【例 7.7】（实例文件：ch07\7.7.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>背景图片：</h4>
<table border="1"
background="images/1.gif">
<tr>
  <td>100</td>
  <td>200</td>
</tr>
<tr>
  <td>300</td>
  <td>400</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-8 所示。

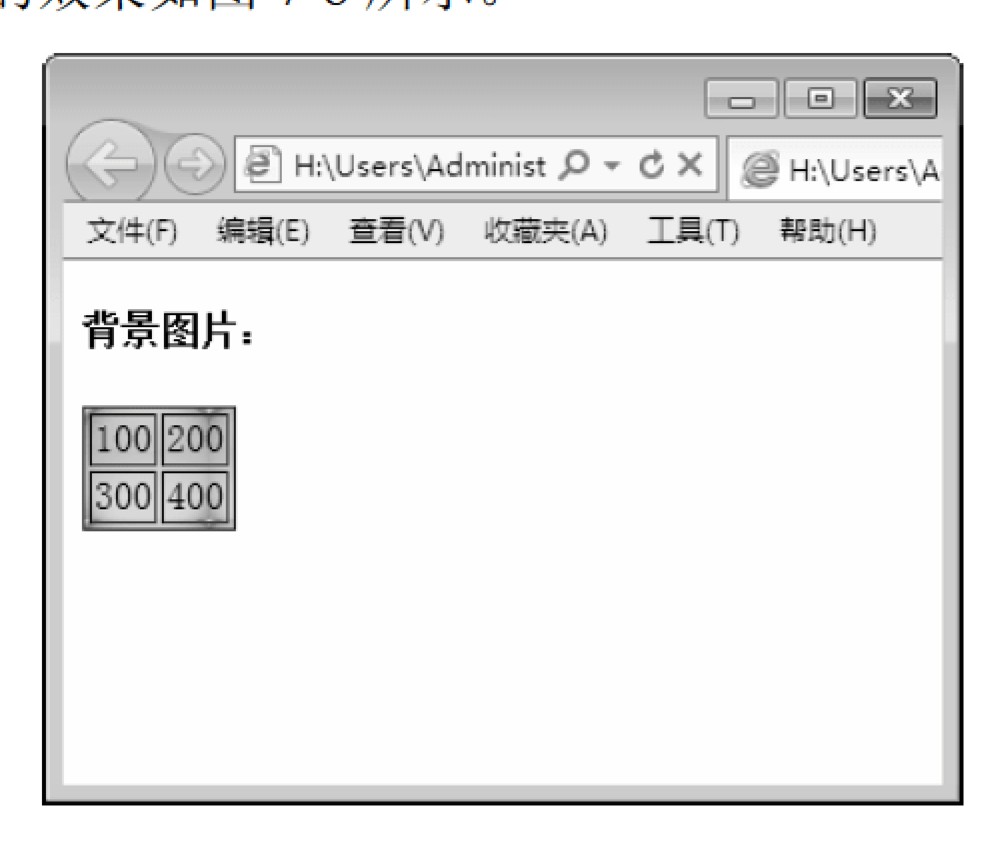


图 7-8　程序运行结果

7.2.5　设置单元格背景

除了可以为表格设置背景外，还可以为单元格设置背景，例如为单元格添加背景。

【例 7.8】（实例文件：ch07\7.8.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>单元格背景</h4>
```

```
<table border="1">
<tr>
  <td bgcolor="red">100000</td>
  <td>200000</td>
</tr>
<tr>
  <td background="images/1.gif">200000</td>
  <td>300000</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-9 所示。



图 7-9　程序运行结果

7.2.6　合并单元格

在实际应用中，并非所有表格都是规范的几行几列，而是需要将某些单元格进行合并，以符合某种内容上的需要。在 HTML 中合并的方向有两种，一种是上下合并，一种是左右合并，这两种合并方式只需要使用 td 标记的两个属性。

1．用 colspan 属性合并左右单元格

左右单元格的合并需要使用 td 标记的 colspan 属性完成，格式如下：

```
<td colspan="数值">单元格内容</td>
```

其中，colspan 属性的取值为数值型整数，代表几个单元格进行左右合并。

例如，在上面表格的基础上，将 A1 和 B1 单元格合并成一个单元格。为第一行的第一个<td>标记增加 colspan="2"属性，并且将 B1 单元格的<td>标记删除。

【例 7.9】（实例文件：ch07\7.9.html）

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>单元格左右合并</title>
</head>
<body>
<table border="1">
 <tr>
   <td colspan="2">A1 B1</td>
   <td>C1</td>
 </tr>
 <tr>
   <td>A2</td>
   <td>B2</td>
   <td>C2</td>
 </tr>
 <tr>
   <td>A3</td>
   <td>B3</td>
   <td>C3</td>
 </tr>
 <tr>
   <td>A4</td>
   <td>B4</td>
   <td>C4</td>
 </tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-10 所示。



图 7-10 单元格左右合并

从预览图中可以看到，A1 和 B1 单元格合并成了一个单元格，C1 还在原来的位置上。



注 意

合并单元格以后，相应的单元格标记就应该减少，例如，A1 和 B1 合并后，B1 单元格的<td></td>标记就应该丢掉，否则单元格就会多出一个，并且后面单元格依次向右位移。

2. 用 rowspan 属性合并上下单元格

上下单元格的合并需要为<td>标记增加 rowspan 属性，格式如下：

```
<td rowspan="数值">单元格内容</td>
```

其中，rowspan 属性的取值为数值型整数，代表几个单元格进行上下合并。

例如，在上面表格的基础上，将 A1 和 A2 单元格合并成一个单元格。为第一行的第一个<td>标记增加 rowspan="2"属性，并且将 A2 单元格的<td>标记删除。

【例 7.10】（实例文件：ch07\7.10.html）

```
<!DOCTYPE html>
<html>
<head>
<title>单元格左右合并</title>
</head>
<body>
<table border="1">
  <tr>
     <td rowspan="2">A1</td>
     <td>B1</td>
     <td>C1</td>
  </tr>
  <tr>
     <td>B2</td>
     <td>C2</td>
  </tr>
  <tr>
     <td>A3</td>
     <td>B3</td>
     <td>C3</td>
  </tr>
  <tr>
     <td>A4</td>
     <td>B4</td>
     <td>C4</td>
```

```
  </tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-11 所示。



图 7-11 单元格上下合并

从预览图中可以看到，A1 和 A2 单元格合并成了一个单元格。

通过上面对左右单元格合并和上下单元格合并的操作，读者会发现，合并单元格就是“丢掉”某些单元格。对于左右合并，就是以左侧为准，将右侧要合并的单元格“丢掉”；对于上下合并，就是以上侧为准，将下侧要合并的单元格“丢掉”。如果一个单元格既要向右合并，又要向下合并，该如何实现呢？

【例 7.11】（实例文件：ch07\7.11.html）

```
<!DOCTYPE html>
<html>
<head>
<title>单元格左右合并</title>
</head>
<body>
<table border="1">
  <tr>
    <td colspan="2" rowspan="2">A1B1<br>A2B2</td>
    <td>C1</td>
  </tr>
  <tr>
    <td>C2</td>
  </tr>
  <tr>
    <td>A3</td>
```

```
        <td>B3</td>
        <td>C3</td>
    </tr>
    <tr>
        <td>A4</td>
        <td>B4</td>
        <td>C4</td>
    </tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-12 所示。

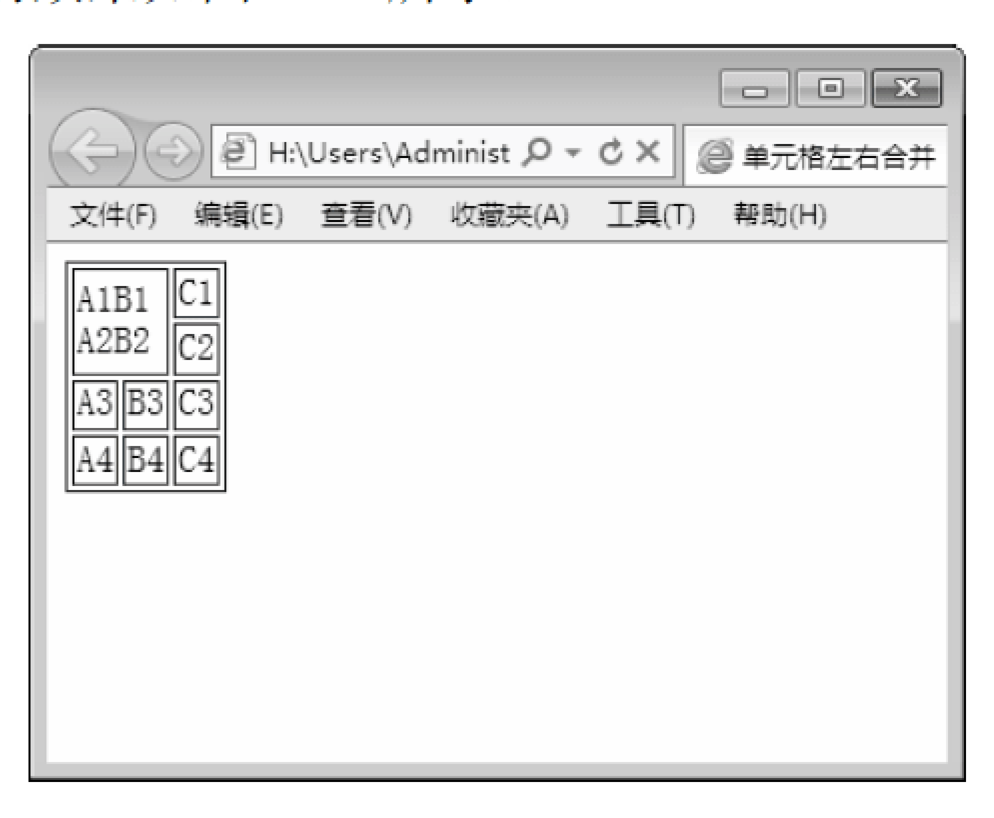


图 7-12　两个方向合并单元格

从上面的代码可以看到，A1 单元格向右合并 B1 单元格，向下合并 A2 单元格，并且 A2 单元格向右合并 B2 单元格。

3．使用 Dreamweaver CS5.5 合并单元格

使用 HTML 创建表格非常麻烦，在 Dreamweaver CS5.5 工具中，提供了表格的快捷操作，类似于在 Word 工具中编辑表格的操作。在 Dreamweaver CS5.5 中创建表格，只需要单击“插入”菜单下的“表格”命令，在出现的对话框中指定表格的行数、列数、宽度和边框中，即可在光标处创建一个空白表格。选择表格之后，属性面板提供了表格的常用操作，如图 7-13 所示。

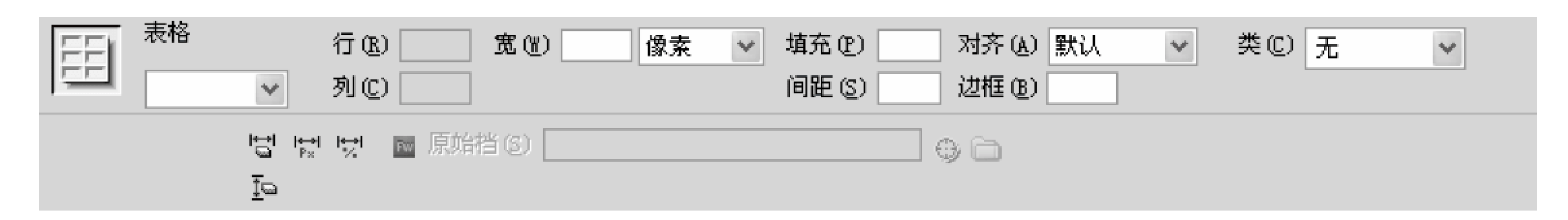


图 7-13　表格属性面板



将鼠标悬停于按钮上，数秒之后会出现命令提示。

关于表格的操作不在赘述，请读者自行操作，这里重点讲解如何使用 Dreamweaver CS5.5 合并单元格。在 Dreamweaver CS5.5 可视化操作中，提供了合并与拆分单元格两种操作。拆分单元格的操作，其实还是进行的合并操作。进行单元格合并和拆分时，请将光标置于单元格内，如果选择了一个单元格，拆分命令有效，如图 7-14 所示。如果选择了两个或两个以上单元格，合并命令有效。

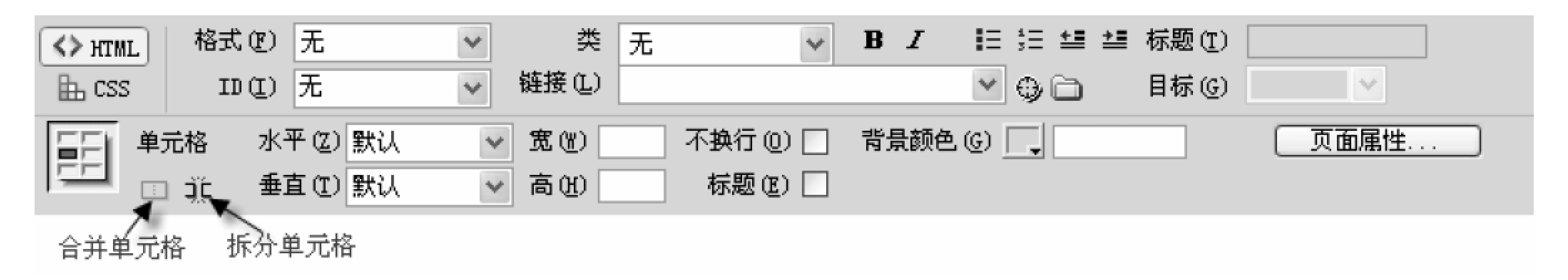


图 7-14　拆分单元格有效

7.2.7　排列单元格中的内容

使用 align 属性可以排列单元格内容，以便创建一个美观的表格。

【例 7.12】（实例文件：ch07\7.12.html）

```
<!DOCTYPE html>
<html>
<body>
<table width="400" border="1">
<tr>
  <th align="left">项目</th>
  <th align="right">一月</th>
  <th align="right">二月</th>
</tr>
<tr>
  <td align="left">衣服</td>
  <td align="right">$241.10</td>
  <td align="right">$50.20</td>
</tr>
<tr>
  <td align="left">化妆品</td>
  <td align="right">$30.00</td>
  <td align="right">$44.45</td>
</tr>
<tr>
  <td align="left">食物</td>
```

```
  <td align="right">$730.40</td>
  <td align="right">$650.00</td>
</tr>
<tr>
  <th align="left">总计</th>
  <th align="right">$1001.50</th>
  <th align="right">$744.65</th>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-15 所示。

项目	一月	二月
衣服	$241.10	$50.20
化妆品	$30.00	$44.45
食物	$730.40	$650.00
总计	**$1001.50**	**$744.65**

图 7-15

7.2.8　设置单元格的行高与列宽

使用 Cell padding 来创建单元格内容与其边框之间的空白，从而调整表格的行高与列宽。

【例 7.13】（实例文件：ch07\7.13.html）

```
<!DOCTYPE html>
<html>
<body>
<h4>调整前</h4>
<table border="1">
<tr>
  <td>1000</td>
  <td>2000</td>
</tr>
<tr>
  <td>2000</td>
  <td>3000</td>
```

```
</tr>
</table>
<h4>调整后</h4>
<table border="1"
cellpadding="10">
<tr>
 <td>1000</td>
 <td>2000</td>
</tr>
<tr>
 <td>2000</td>
 <td>3000</td>
</tr>
</table>
</body>
</html>
```

在 IE 9.0 中预览网页的效果如图 7-16 所示。

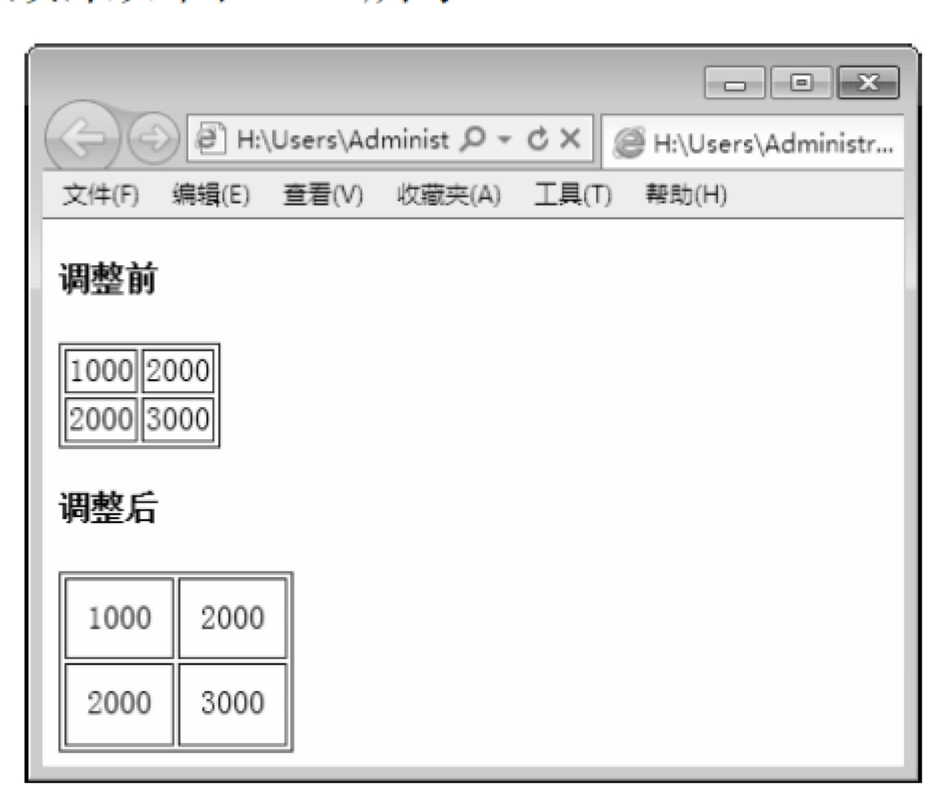


图 7-16 程序运行结果

7.3 完整的表格标记

上面讲述了表格中最常用也是最基本的三个标记<table>、<tr>和<td>，使用它们可以构建出最简单的表格。为了让表格结构更清楚，以及配合后面学习的 CSS 样式，可更方便地制作各种表格，还会出现表头、主体、脚注等。

按照表格结构，可以把表格的行分组，称为“行组”。不同的行组具有不同的意义。行组分为三类——“表头”、“主体”和“脚注”。三者相应的 HTML 标记依次为<thead>、<tbody>和<tfoot>。

此外，在表格中还有两个标记。标记<caption>表示表格的标题。在一行中，除了<td>标记表示一个单元格以外，还可以使用<th>表示该单元格是这一行的“行头”。

【例 7.14】（实例文件：ch07\7.14.html）

```
<!DOCTYPE html>
<html>
<head>
<title>完整表格标记</title>
<style>
tfoot{
        background-color:#FF3;
}
</style>
</head>
<body>
<table border="1">
  <caption>学生成绩单</caption>
  <thead>
      <tr>
          <th>姓名</th><th>性别</th><th>成绩</th>
      </tr>
  </thead>
    <tfoot>
      <tr>
          <td>平均分</td><td colspan="2">540</td>
      </tr>
  </tfoot>
  <tbody>
      <tr>
          <td>张三</td><td>男</td><td>560</td>
      </tr>
      <tr>
          <td>李四</td><td>男</td><td>520</td>
      </tr>
  </tbody>
</table>
</body>
</html>
```

从上面的代码可以发现，使用 caption 表格定义了表格标题，<thead>、<tbody>和<tfoot>标记对表格进行了分组。在<thead>部分使用<th>标记代替<td>标记定义单元格，<th>标记定义的单元格默认加粗，网页预览的效果如图 7-17 所示。



图 7-17　完整的表格结构



<caption>标签必须紧随<table>标签之后。

7.4　综合实例——制作计算机报价表

利用所学的表格知识，制作如图 7-18 所示的计算机报价表。

计算机报价单

型号	类型	价格	图片
宏碁 (Acer) AS4552-P362G32MNCC	笔记本	￥2799	
戴尔 (Dell) 14VR-188	笔记本	￥3499	
联想 (Lenovo) G470AH2310W42G500P7CW3(DB)-CN	笔记本	￥4149	
戴尔家用 (DELL) I560SR-656	台式	￥3599	
宏图奇眩(Hiteker) HS-5508-TF	台式	￥3399	
联想 (Lenovo) G470	笔记本	￥4299	

图 7-18　计算机报价单

具体操作步骤如下：

01 新建 HTML 文档，并对其简化，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>完整表格标记</title>
</head>
<body>
</body>
</html>
```

02 保存 HTML 文件，选择相应的保存位置，文件名为“综合示例——购物简易计算器.html”。

03 在 HTML 文档的 body 部分增加表格及内容，代码如下：

```
<table>
  <caption>计算机报价单</caption>
   <tr>
     <th>型号</th>
     <th>类型</th>
     <th>价格</th>
     <th>图片</th>
   </tr>
   <tr>
     <td>宏碁 (Acer) AS4552-P362G32MNCC</td>
     <td>笔记本</td>
     <td>￥2799</td>
     <td><img src="images/Acer.jpg" width="120" height="120"></td>
   </tr>
   <tr>
     <td>戴尔 (Dell) 14VR-188</td><td>笔记本</td>
     <td>￥3499</td>
     <td><img src="images/Dell.jpg" width="120" height="120"></td>
   </tr>
    <tr>
     <td>联想 (Lenovo) G470AH2310W42G500P7CW3(DB)-CN </td>
     <td>笔记本</td>
     <td>￥4149</td>
     <td><img src="images/Lenovo.jpg" width="120" height="120"></td>
   </tr>
```

```
    <tr>
      <td>戴尔家用（DELL） I560SR-656</td>
      <td>台式</td>
      <td>￥3599</td>
      <td><img src="images/DellT.jpg" width="120" height="120"></td>
    </tr>
    <tr>
      <td>宏图奇眩(Hiteker)  HS-5508-TF</td>
      <td>台式</td>
      <td>￥3399</td>
      <td><img src="images/Hiteker.jpg" width="120" height="120"></td>
    </tr>
    <tr>
      <td>联想（Lenovo) G470</td>
      <td>笔记本</td>
      <td>￥4299</td>
      <td><img src="images/LenovoG.jpg" width="120" height="120"></td>
    </tr>
</table>
```



利用 caption 标记制作表格的标题，<th>代替<td>作为标题行单元格。可以将图片放在单元格内，即在<td>标记内使用<img>标记。

04 在 HTML 文档的 head 部分，增加 CSS 样式，为表格增加边框及相应的修饰，代码如下：

```
<style>
table{
     /*表格增加线宽为 3 的橙色实线边框*/
     border:3px solid #F60;
}
caption{
     /*表格标题字号 36*/
     font-size:36px;
}
th,td{
     /*表格单元格（th、td）增加边线*/
     border:1px solid #F90;
}
</style>
```

05 保存网页后，即可查看最终效果。

7.5　问题解答

1．在 Dreamweaver CS5.5 中如何选择多个单元格？

在 Dreamweaver CS5.5 中选择单元格的操作类似于文字处理工具 Word，按下鼠标左键拖动鼠标，经过的单元格都会被选择。按下 Ctrl 键，单击某个单元格，该单元格将会被选择，这些单元格可以是连续的也可以是不连续的。在需要选择区域的开头单元格中单击，按下 Shift 键，在区域的末尾单元格中单击，开头和结尾单元格组成的区域内的所有单元格将会被选择。

2．表格除了显示数据，还可以进行布局，为何不使用表格进行布局？

在互联网刚刚开始普及时，网页非常简单，形式也非常单调，当时美国设计 David Siegel 发明了表格布局，并风靡全球。在表格布局的页面中，表格不但需要显示内容，还要控制页面的外观及显示位置，导致页面代码过多，结构与内容无法分离，给网站的后期维护和很多其他方面带来了麻烦。

3．使用<thead>、<tbody>和<tfoot>标记对行进行分组的意义何在？

在 HTML 文档中增加<thead>、<tbody>和<tfoot>标记虽然从外观上不能看出任何变化，但是它们却使文档的结构更加清晰。使用<thead>、<tbody>和<tfoot>标记除了使文档更加清晰之外，还有一个更重要的意义，方便使用 CSS 样式对表格的各个部分进行修饰，从而可制作出更出色的表格。

第 8 章　使用 HTML 5 创建表单

在网页中，表单的作用比较重要，主要负责采集浏览者的相关数据。例如常见的注册表、调查表和留言表等。在 HTML 5 中，表单拥有多个新的表单输入类型，这些新特性提供了更好的输入控制和验证。

8.1　表单概述

表单主要用于收集网页上浏览者的相关信息。其标签为<form></form>。表单的基本语法格式如下：

```
<form action="url" method="get|post" enctype="mine">
</form >
```

其中，action="url" 指定处理提交表单的格式，可以是 URL 地址或电子邮件地址。method="get/post"指明提交表单的 HTTP 方法。enctype=mine 指明用来把表单提交给服务器时的互联网媒体形式。

表单是一个能够包含表单元素的区域。通过添加不同的表单元素，将显示不同的效果。

【例 8.1】（实例文件：ch08\8.1.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
下面是输入用户登录信息
<br>
用户名称
<input type="text" name="user">
<br>
用户密码
<input type="password" name="password">
<br>
<input type="submit" value="登录">
</form>
</body>
</html>
```

在 IE 9.0 中浏览效果如图 8-1 所示，可以看到用户登录信息页面。

图 8-1　用户登录窗口

8.2　表单基本元素的使用

表单元素是能够让用户在表单中输入信息的元素。常见的有文本框、密码框、下拉菜单、单选框和复选框等。

8.2.1　单行文本输入框

文本框是一种让访问者自己输入内容的表单对象，通常被用来填写单个字或者简短的回答，例如用户姓名和地址等。代码格式如下：

```
<input type="text" name="..." size="..." maxlength="..." value="...">
```

其中，type="text"定义单行文本输入框，name 属性定义文本框的名称，要保证数据的准确采集，必须定义一个独一无二的名称；size 属性定义文本框的宽度，单位是单个字符宽度；maxlength 属性定义最多输入的字符数；value 属性定义文本框的初始值。

【例 8.2】（实例文件：ch08\8.2.html）

```
<!DOCTYPE html>
<html>
<head><title>输入用户的姓名</title></head>
<body>
<form>
请输入您的姓名：
<input type="text" name="yourname" size="20" maxlength="15">
请输入您的地址：
<input type="text" name="youradr" size="20" maxlength="15">
```

```
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-2 所示，可以看到两个单行文本输入框。



图 8-2 单行文本输入框

8.2.2 多行文本输入框

多行输入框（textarea）主要用于输入较长的文本信息。代码格式如下：

```
<textarea name="..." cols="..." rows="..." wrap="..."></textarea >
```

其中，name 属性定义多行文本框的名称，要保证数据的准确采集，必须定义一个独一无二的名称；cols 属性定义多行文本框的宽度，单位是单个字符宽度；rows 属性定义多行文本框的高度，单位是单个字符宽度；wrap 属性定义输入内容大于文本域时的显示方式。

【例 8.3】（实例文件：ch08\8.3.html）

```
<!DOCTYPE html>
<html>
<head><title>多行文本输入</title></head>
<body>
<form>
请输入您最新的工作情况<br>
<textarea name="yourworks" cols ="50" rows = "5"></textarea>
<br>
<input type="submit" value="提交">
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-3 所示，可以看到多行文本输入框。

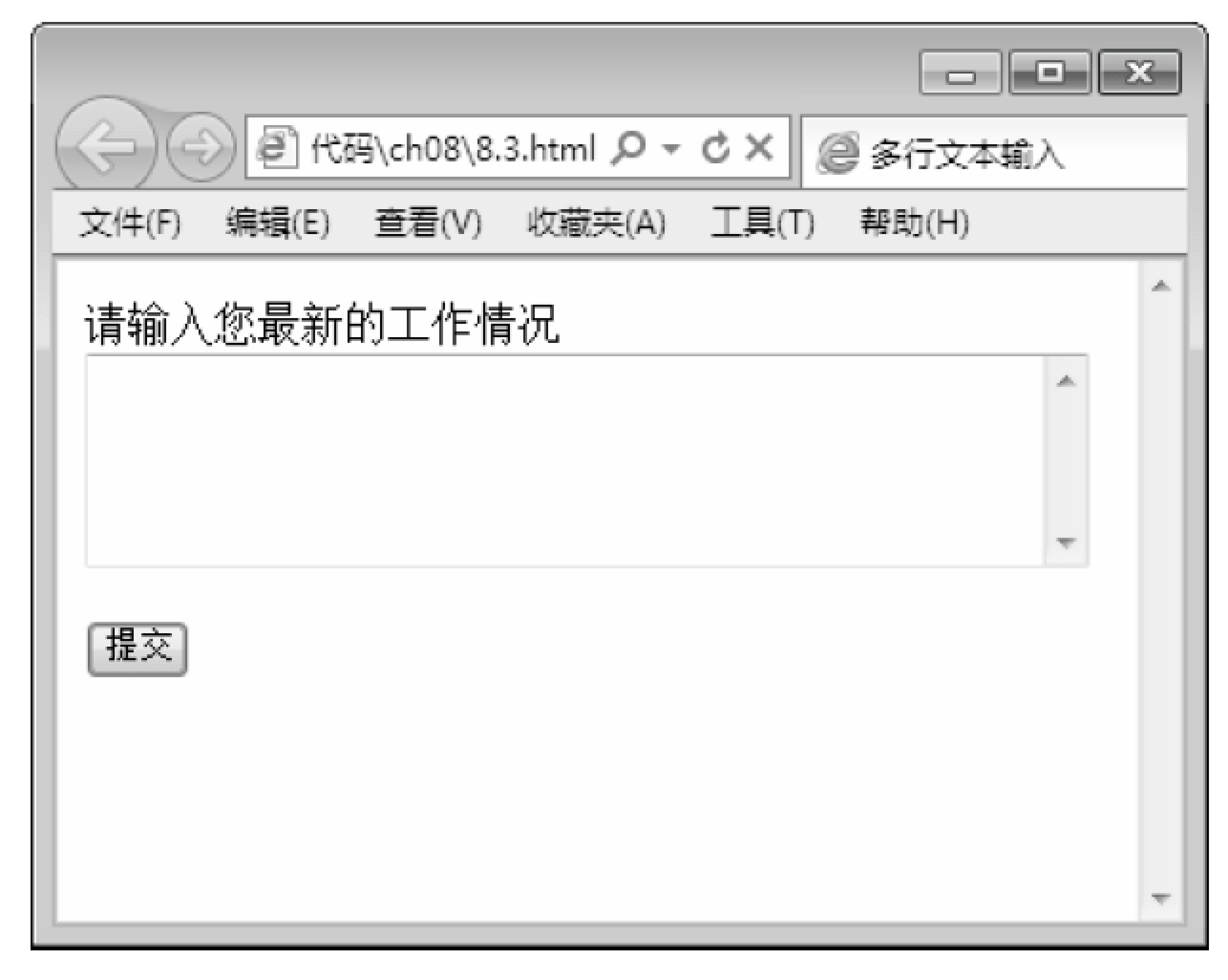


图 8-3　多行文本输入框

8.2.3　密码域

密码输入框是一种特殊的文本域，主要用于输入保密信息。当网页浏览者输入文本时，显示的是黑点或者其他符号，这样就增加了输入文本的安全性。代码格式如下：

```
<input type="password" name="..." size="..." maxlength="...">
```

其中 type="password"定义密码框；name 属性定义密码框的名称，要保证唯一性；size 属性定义密码框的宽度，单位是单个字符宽度；maxlength 属性定义最多输入的字符数。

【例 8.4】（实例文件：ch08\8.4.html）

```
<!DOCTYPE html>
<html>
<head><title>输入用户姓名和密码</title></head>
<body>
<form >
用户姓名：
<input type="text" name="yourname">
<br>
登录密码：
<input type="password" name="yourpw"><br>
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-4 所示，输入用户名和密码时可以看到密码以黑点形式显示。



图 8-4　密码输入框

8.2.4　单选按钮

单选按钮主要是控制网页浏览者在一组选项里只能选择一个选项。代码格式如下：

```
<input type="radio" name=" " value = " ">
```

其中 type="radio"定义单选按钮；name 属性定义单选按钮的名称，单选按钮都是以组为单位使用的，在同一组中的单选项都必须用同一个名称；value 属性定义单选按钮的值，在同一组中，它们的域值必须是不同的。

【例 8.5】（实例文件：ch08\8.5.html）

```
<!DOCTYPE html>
<html>
<head><title>选择感兴趣的图书</title></head>
<body>
<form >
请选择您感兴趣的图书类型：
<br>
<input type="radio" name="book" value = "Book1">网站编程<br>
<input type="radio" name="book" value = "Book2">办公软件<br>
<input type="radio" name="book" value = "Book3">设计软件<br>
<input type="radio" name="book" value = "Book4">网络管理<br>
<input type="radio" name="book" value = "Book5">黑客攻防<br>
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-5 所示，即可看到 5 个单选按钮，用户只能选择其中一个单选按钮。



图 8-5　单选按钮

8.2.5　复选框

复选框主要是让网页浏览者在一组选项里可以同时选择多个选项。每个复选框都是一个独立的元素，都必须有一个唯一的名称。代码格式如下：

```
<input type="checkbox" name=" " value ="">
```

其中 type="checkbox"定义复选框；name 属性定义复选框的名称，在同一组中的复选框都必须用同一个名称；value 属性定义复选框的值。

【例 8.6】（实例文件：ch08\8.6.html）

```
<!DOCTYPE html>
<html>
<head><title>选择感兴趣的图书</title></head>
<body>
<form >
请选择您感兴趣的图书类型：<br>
<input type="checkbox" name="book" value = "Book1">网站编程<br>
<input type="checkbox" name="book" value = "Book2">办公软件<br>
<input type="checkbox" name="book" value = "Book3">设计软件<br>
<input type="checkbox" name="book" value = "Book4">网络管理<br>
<input type="checkbox" name="book" value = "Book5" checked>黑客攻防<br>
</form>
</body>
</html>
```



checked 属性主要用来设置默认选中选项。

在 IE 9.0 中的浏览效果如图 8-6 所示，可看到有 5 个复选框，其中“黑客攻防”复选框默认被选中。

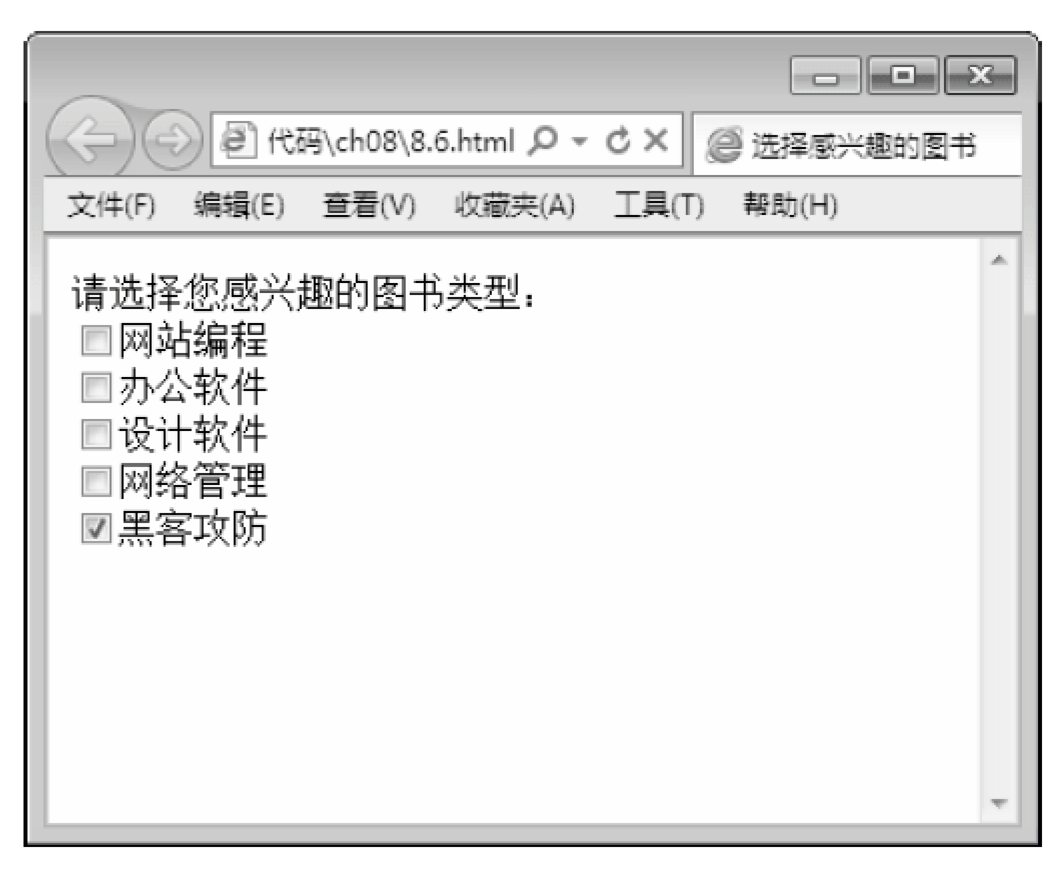


图 8-6　复选框的效果

8.2.6　下拉选择框

下拉选择框主要用于在有限的空间里设置多个选项。下拉选择框既可以用做单选，也可以用做复选。代码格式如下：

```
<select name="..." size="..." multiple>
<option value="..." selected>
...
</option>
 ...
</select>
```

其中 size 属性定义下拉选择框的行数；name 属性定义下拉选择框的名称；multiple 属性表示可以多选，如果不设置本属性，那么只能单选；value 属性定义选择项的值；selected 属性表示默认已经选择本选项。

【例 8.7】（实例文件：ch08\8.7.html）

```
<!DOCTYPE html>
<html>
<head><title>选择感兴趣的图书</title></head>
<body>
<form>
请选择您感兴趣的图书类型：<br>
<select name="fruit" size = "3" multiple>
```

```
<option value="Book1">网站编程
<option value="Book2">办公软件
<option value="Book3">设计软件
<option value="Book4">网络管理
<option value="Book5">黑客攻防
</select>
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-7 所示，即可看到下拉选择框，其中显示为三行选项，用户可以按住 Ctrl 键，选择多个选项。



图 8-7　下拉选择框的效果

8.2.7　普通按钮

普通按钮用来控制其他定义了处理脚本的处理工作。代码格式如下：

```
<input type="button" name="..." value="..." onClick="...">
```

其中 type="button"定义普通按钮；name 属性定义普通按钮的名称；value 属性定义按钮的显示文字；onClick 属性表示单击行为，也可以是其他的事件，通过指定脚本函数来定义按钮的行为。

【例 8.8】（实例文件：ch08\8.8.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
单击下面的按钮，把文本框 1 的内容拷贝到文本框 2 中：
<br/>
```

```
文本框 1:<input type="text" id="field1" value="学习 HTML 5 的技巧">
<br/>
文本框 2:<input type="text" id="field2">
<br/>
<input type="button" name="..." value="单击我" onClick="document.getElementById('field2').value=
document.getElementById('field1').value">
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-8 所示，单击“单击我”按钮，即可实现将文本框中内容复制到文本框 2 中。



图 8-8 单击按钮后的复制效果

8.2.8 提交按钮

提交按钮用来将输入的信息提交到服务器。代码格式如下：

```
<input type="submit" name="..." value="...">
```

其中 type="submit"定义提交按钮；name 属性定义提交按钮的名称；value 属性定义按钮的显示文字。通过提交按钮可以将表单里的信息提交给表单中 action 所指向的文件。

【例 8.9】（实例文件：ch08\8.9.html）

```
<!DOCTYPE html>
<html>
<head><title>输入用户名信息</title></head>
<body>
<form action="http://www.yinhangit.com/yonghu.asp" method="get">
请输入你的姓名：
```

```
<input type="text" name="yourname">
<br>
请输入你的住址：
<input type="text" name="youradr">
<br>
请输入你的单位：
<input type="text" name="yourcom">
<br>
请输入你的联系方式：
<input type="text" name="yourcom">
<br>
<input type="submit" value="提交">
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-9 所示，输入内容后单击“提交”按钮，即可实现将表单中的数据发送到制定的文件。

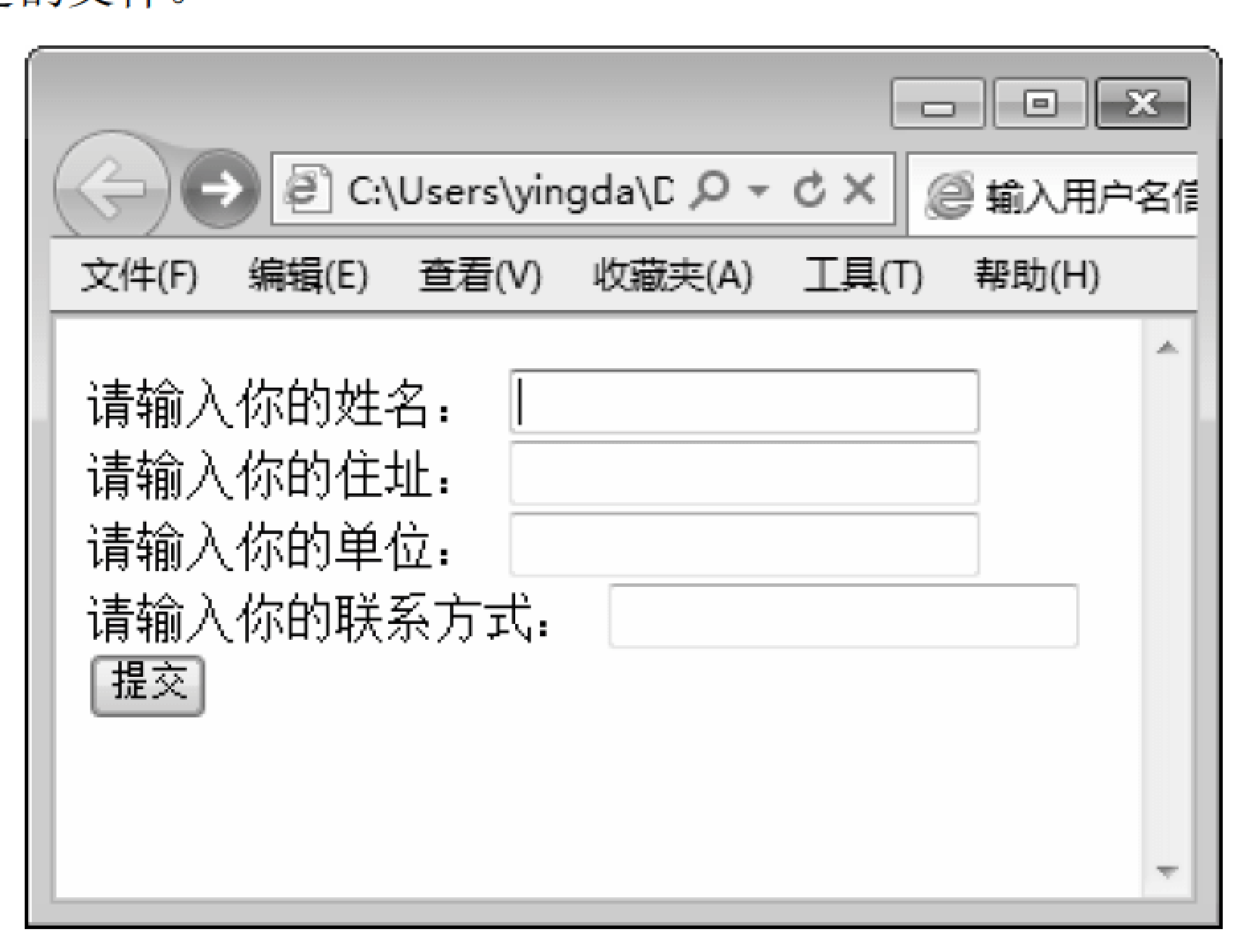


图 8-9　提交按钮

8.2.9　重置按钮

重置按钮用来重置表单中输入的信息。代码格式如下：

```
<input type="reset" name="..." value="...">
```

其中 type="reset"定义复位按钮；name 属性定义复位按钮的名称；value 属性定义按钮的显示文字。

【例 8.10】（实例文件：ch08\8.10.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
请输入用户名称：
<input type='text'>
<br/>
请输入用户密码：
<input type='password'>
<br>
<input type="submit" value="登录">
<input type="reset" value="重置">
</form>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-10 所示，输入内容后单击“重置”按钮，即可实现将表单中的数据清空的目的。

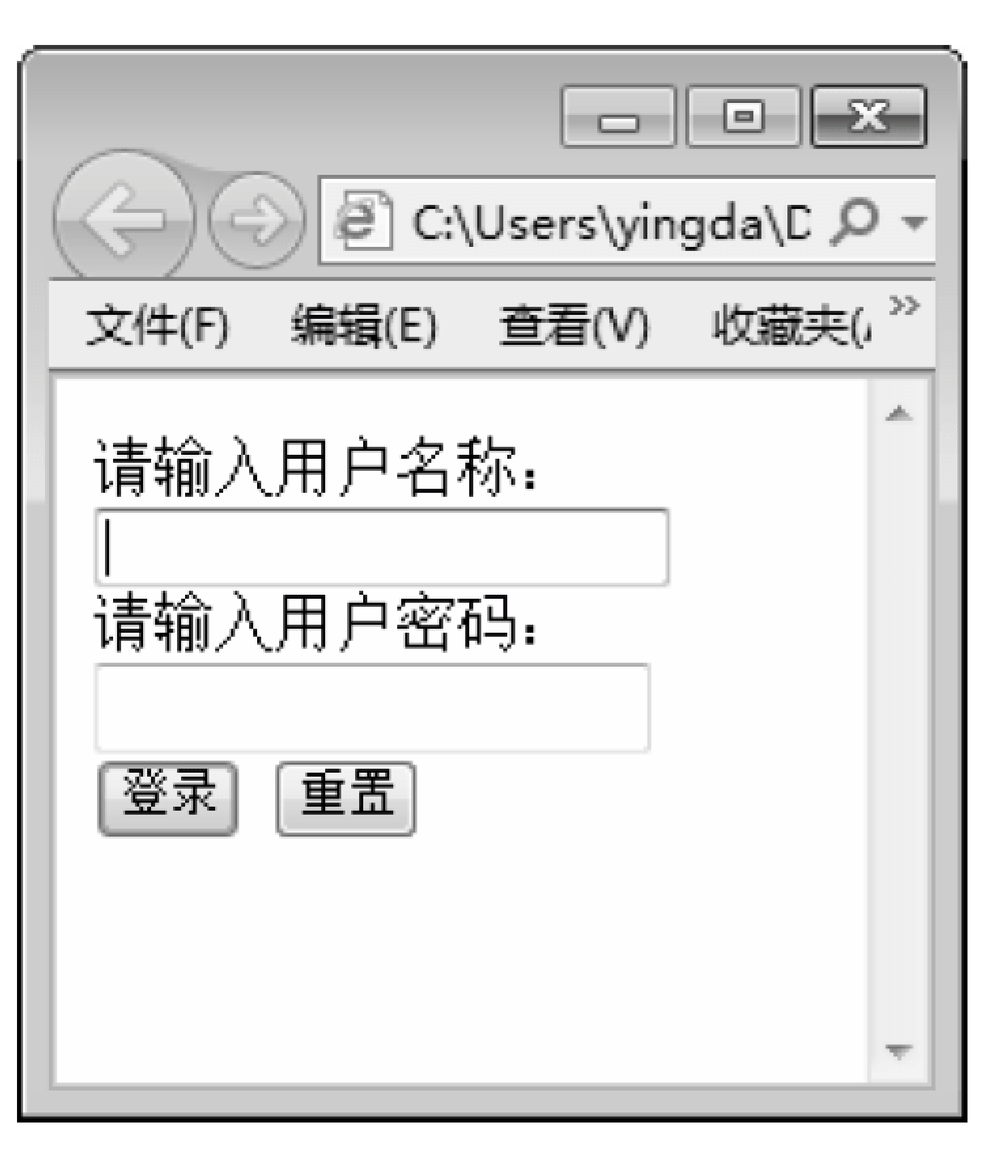


图 8-10 “重置”按钮

8.3 表单高级元素的使用

除了上述基本元素外，HTML 5 中还有一些高级元素。包括 URL、email、time、range、search 等等。对于这些高级属性，IE 9.0 浏览器暂时还不支持，下面将用 Opera 11.60 浏览器查看效果。

8.3.1　url 属性

url 属性用于说明网站网址的，显示为一个文本字段，用于输入 URL 地址。在提交表单时，会自动验证 url 的值。代码格式如下：

```
<input type="url" name="userurl"/>
```

另外，用户可以使用普通属性设置 url 输入框，例如可以使用 max 属性设置其最大值、min 属性设置其最小值、step 属性设置合法的数字间隔、利用 value 属性规定其默认值。对于另外的高级属性中同样的设置不再重复讲述。

【例 8.11】（实例文件：ch08\8.11.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
<br/>
请输入网址：
<input type="url" name="userurl"/>
</form>
</body>
</html>
```

在 Opera 11.60 中的浏览效果如图 8-11 所示，用户可输入相应的网址。



图 8-11　url 属性的效果

8.3.2　email 属性

与 url 属性类似，email 属性用于输入 e-mail 地址。在提交表单时，会自动验证 email 域

的值。代码格式如下：

```
<input type="email" name="user_email"/>
```

【例 8.12】（实例文件：ch08\8.12.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
<br/>
请输入您的邮箱地址：
<input type="email" name="user_email"/>
<br>
<input type="submit" value="提交">
</form>
</body>
</html>
```

在 Opera 11.60 中的浏览效果如图 8-12 所示，用户可输入相应的邮箱地址。如果用户输入的邮箱地址不合法，单击“提交”按钮后会弹出图 8-12 中的提示信息。



图 8-12　eamil 属性的效果

8.3.3　date 和 time

在 HTML 5 中，新增了日期和时间输入类型，包括 date、datetime、datetime-local、month、week 和 time。它们的具体含义如下表所示。

属性	含义
date	选取日、月、年
month	选取月、年
week	选取周和年
time	选取时间
datetime	选取时间、日、月、年
datetime-local	选取时间、日、月、年（本地时间）

上述属性的代码格式类似，例如以 date 属性为例，代码格式如下：

```
<input type="date" name="user_date" />
```

【例 8.13】（实例文件：ch08\8.13.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
<br/>
请选择购买商品的日期：
<br>
<input type="date" name="user_date" />
</form>
</body>
</html>
```

在 Opera 11.6 中的浏览效果如图 8-13 所示，用户单击输入框中的向下按钮，即可在弹出的窗口中选择需要的日期。



图 8-13　date 属性的效果

8.3.4 number 属性

number 属性提供了一个输入数字的输入类型。用户可以直接输入数字或者通过单击微调框中的向上或者向下按钮选择数字。代码格式如下：

```
<input type="number" name="shuzi" />
```

【例 8.14】（实例文件：ch08\8.14.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
<br/>
此网站我曾经来
<input type="number" name="shuzi "/>次了哦！
</form>
</body>
</html>
```

在 Opera 11.6 中的浏览效果如图 8-14 所示，用户可以直接输入数字，也可以单击微调按钮选择合适的数字。



图 8-14　number 属性的效果



强烈建议用户使用 min 和 max 属性规定输入的最小值和最大值。

8.3.5　range 属性

range 属性用来显示滚动的控件。和 number 属性一样，用户可以使用 max、min 和 step 属性控制控件的范围。代码格式如下：

```
<input type="range" name="" min="" max="" />
```

其中 min 和 max 分别控制滚动控件的最小值和最大值。

【例 8.15】（实例文件：ch08\8.15.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
<br/>
英语成绩公布了！我的成绩名名次为：
<input type="range" name="ran" min="1" max="10" />
</form>
</body>
</html>
```

在 Opera 11.6 中的浏览效果如图 8-15 所示，用户可以拖曳滑块，从而选择合适的数字。



图 8-15　range 属性的效果



默认情况下，滑块位于滚珠的中间位置。如果用户指定的最大值小于最小值，则允许使用反向滚动轴，目前浏览器对这一属性还不能很好地支持。

8.3.6 required 属性

required 属性规定必须在提交之前填写输入域（不能为空）。required 属性适用于以下类型的输入属性：text、search、url、email、password、date、pickers、number、checkbox 和 radio 等。

【例 8.16】（实例文件：ch08\8.16.html）

```
<!DOCTYPE html>
<html>
<body>
<form>
下面是输入用户登录信息
<br>
用户名称
<input type="text" name="user" required="required">
<br>
用户密码
<input type="password" name="password" required="required">
<br>
<input type="submit" value="登录">
</form>
</body>
</html>
```

在 Opera 11.6 中的浏览效果如图 8-16 所示，用户如果只是输入密码，然后单击“登录”按钮，将弹出提醒信息。



图 8-16　required 属性的效果

8.4　综合实例——创建用户反馈表单

在本实例中，将使用表单内的各种元素来开发一个简单网站的用户意见反馈页面。

反馈表单非常简单，通常包含三个部分，需要在页面上方给出标题，标题下方是正文部分，即表单元素，最下方是表单元素提交按钮。在设计这个页面时，需要把“用户注册”标题设置成 H1 大小，正文使用 p 来限制表单元素。

具体操作步骤如下：

构建 HTML 页面，实现表单内容：

```
<!DOCTYPE html>

<html>
<head>
<title>用户反馈页面</title>
</head>
<body>
<h1 align=center>用户反馈表单</h1>
<form method="post" >
<p>姓    名：
<input type="text" class=txt size="12" maxlength="20" name="username" />
</p><p>性    别：
<input type="radio" value="male" />男
<input type="radio" value="female" />女
</p><p>年    龄：
<input type="text" class=txt name="age"   />
</p>
<p>联系电话：
<input type="text" class=txt name="tel" />
</p><p>电子邮件：
<input type="text" class=txt name="email" />
</p><p>联系地址：
<input type="text"   class=txt name="address" />
</p>
<p>
请输入您对网站的建议<br>
<textarea name="yourworks" cols ="50" rows = "5"></textarea>
<br>
<input type="submit" name="submit" value="提交"/>
<input type="reset" name="reset" value="清除" />
</p>
</form>
```

```
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 8-17 所示，可以看到创建了一个用户反馈表单，包含一个标题“用户反馈表单”及“姓名”、“性别”、“年龄”、“联系电话”、“电子邮件”、“联系地址”、“请输入您对网站的建议”等输入框和“提交”按钮。



图 8-17 用户反馈页面

8.5 问题解答

1. 如何在表单中实现文件上传文本框？

在 HTML 5 语言中，使用 file 属性实现文件上传文本框。语法格式为：<input type="file" name="..." size=" " maxlength=" ">。其中 type="file"定义为文件上传框，name 属性为文件上传文本框的名称，size 属性定义文件上传文本框的宽度，单位是单个字符宽度；maxlength 属性定义最多输入的字符数，文件上传框的显示效果如图 8-18 所示。



图 8-18 文件上传框

2. 制作的单选框为什么可以同时选中多个?

此时用户需要检查单选框的名称，保证同一组中的单选框名称必须相同，这样才能保证单选框只能选中一个。

第 9 章　使用 HTML 5 绘制图形

HTML 5 呈现了很多的新特性，这在之前的 HTML 中是不可见到的。其中一个最值得提及的特性就是 HTML Canvas，可以对 2D 或位图进行动态、脚本的渲染。Canvas 是一个矩形区域，使用 JavaScript 可以控制其每一个像素。

9.1　canvas 概述

canvas 是一个新的 HTML 元素，这个元素可以被 Script 语言（通常是 JavaScript）用来绘制图形。例如可以用它来画图、合成图像、或做简单的动画。

9.1.1　添加 canvas 元素

canvas 标签是一个矩形区域，它包含两个属性 width 和 height，分别表示矩形区域的宽度和高度，这两个属性都是可选的，并且都可以通过 CSS 来定义，其默认值是 300 像素和 150 像素。

canvas 在网页中常用形式如下：

```
<canvas id="myCanvas" width="300" height="200" style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
```

上面示例代码中，id 表示画布对象名称，width 和 height 分别表示宽度和高度；最初的画布是不可见的，为了观察这个矩形区域，这里使用了 CSS 样式，即 style 标记。style 表示画布的样式。如果浏览器不支持画布标记，会显示画布中间的提示信息。

画布 canvas 本身不具有绘制图形的功能，只是一个容器，如果读者对 Java 语言非常了解，就会发现 HTML 5 的画布和 Java 中的 Panel 面板非常相似，都可以在容器内绘制图形。既然放好了 canvas 画布元素，就可以使用脚本语言 JavaScript 在网页上绘制图像了。

使用 canvas 结合 JavaScript 绘制图形，一般情况下需要下面几个步骤：

01 JavaScript 使用 id 来寻找 canvas 元素，即可获取当前画布对象，其代码如下：

```
var c=document.getElementById("myCanvas");
```

02 创建 context 对象，其代码如下：

```
var cxt=c.getContext("2d");
```

getContext 方法返回一个指定 contextId 的上下文对象，如果指定的 id 不被支持，则返回

null，当前唯一被强制必须支持的是 2d，也许在将来会有 3d，注意，指定的 id 是大小写敏感的。对象 cxt 建立之后，就可以拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

03 绘制图形：

```
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
```

fillStyle 方法将其染成红色，fillRect 方法规定了形状、位置和尺寸。这两行代码将绘制一个红色的矩形。

9.1.2 绘制矩形

单独的一个 canvas 标记只是在页面中定义了一块矩形区域，并无特别之处，开发人员只有配合使用 JavaScript 脚本，才能够完成各种图形，线条，以及复杂的图形变换操作，与基于 SVG 来实现同样绘图效果比较，canvas 绘图是一种像素级别的位图绘图技术，而 SVG 则是一种矢量绘图技术。

使用 canvas 和 JavaScript 绘制一个矩形，可能会涉及到一个或多个方法，这些方法如下表所示。

方法	功能
fillRect	绘制一个矩形，这个矩形区域没有边框，只有填充色。这个方法有 4 个参数，前两个表示左上角的坐标位置，第三个参数为长度，第四个参数为高度
strokeRect	绘制一个带边框的矩形。该方法的 4 个参数的解释同 fillRect 方法。
clearRect	清除一个矩形区域，被清除的区域将没有任何线条。该方法的 4 个参数的解释同 fillRect 方法。

【例 9.1】（实例文件：ch09\9.1.html）

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="300" height="200" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="rgb(0,0,200)";
cxt.fillRect(10,20,100,100);
</script>
</body>
</html>
```

上面代码中，首先定义画布对象，其 id 名称为 myCanvas，其高度和宽度都为 500 像素，并定义了画布边框显示样式。

在 JavaScript 代码中，首先获取画布对象，然后使用 getContext 获取当前 2d 的上下文对象，并使用 fillRect 绘制矩形。其中涉及到 fillStyle 属性，fillstyle 用于设定填充的颜色、透明度等，如果设置为 rgb（200,0,0），则表示颜色，不透明；如果设为 rgb（0,0,200,0.5），也表示为颜色，透明度为 50%。

在 IE 9.0 中浏览效果如图 9-1 所示，可以看到网页中，在蓝色边框中显示了一个蓝色矩形。

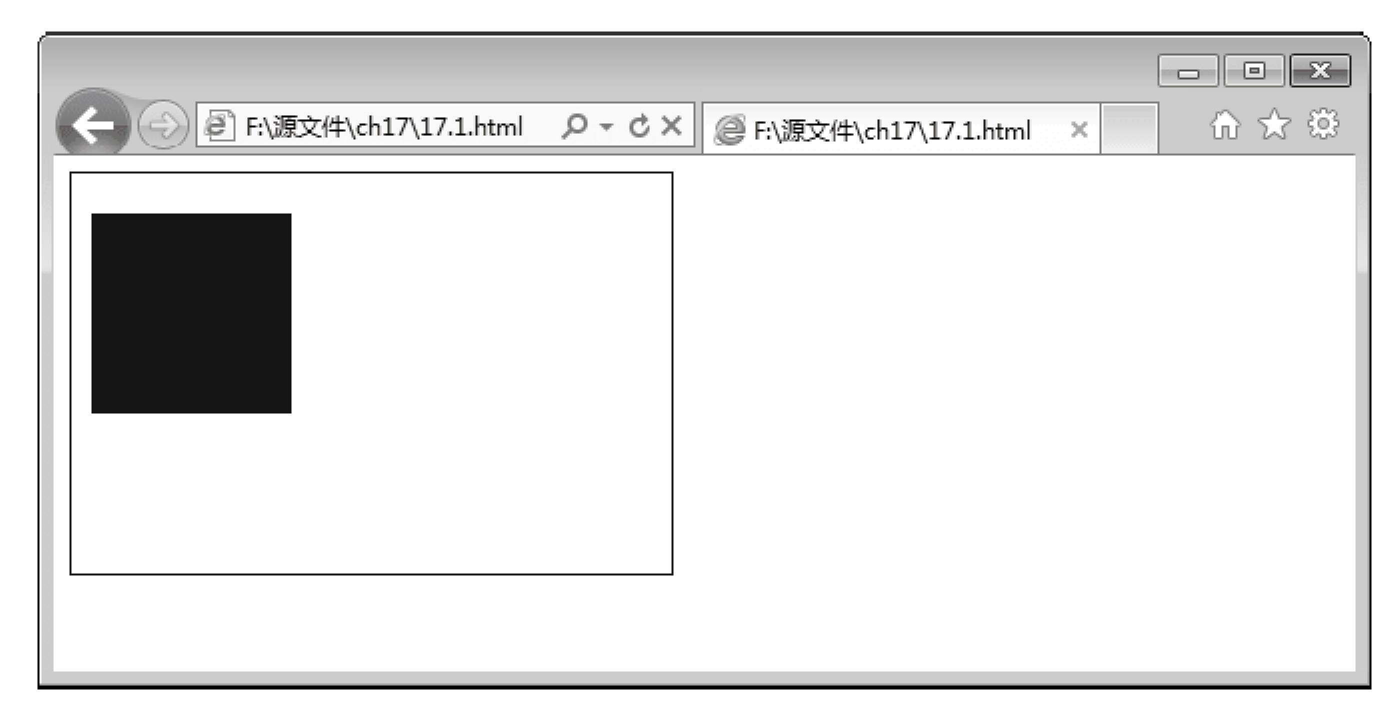


图 9-1　绘制矩形

9.2　绘制基本形状

画布 canvas 结合 JavaScript 不但可以绘制简单的矩形，还可以绘制一些其他的常见图形，例如直线、圆等。

9.2.1　绘制圆形

基于 canvas 的绘图并不是直接在 canvas 标记所创建的绘图画面上进行各种绘图操作，而是依赖画面所提供的渲染上下文（Rendering Context）来操作的，所有的绘图命令和属性都定义在渲染上下文当中。在通过 canvas id 获取相应的 DOM 对象之后首先要做的事情就是获取渲染上下文对象。渲染上下文与 canvas 一一对应，无论对同一 canvas 对象调用几次 getContext() 方法，都将返回同一个上下文对象。

在画布中绘制圆形，可能要涉及到下面几个方法，如下表所示。

方法	功能
beginPath()	开始绘制路径
arc(x,y,radius,startAngle, endAngle,anticlockwise)	x 和 y 定义的是圆的原点，radius 是圆的半径，startAngle 和 endAngle 是弧度，不是度数，anticlockwise 用来定义所画圆的方向，值是 true 或 false
closePath()	结束路径的绘制
fill()	进行填充
stroke()	方法设置边框

路径是绘制自定义图形的好方法，在 canvas 中通过 beginPath()方法开始绘制路径，这个时候就可以绘制直线、曲线等，绘制完成后调用 fill()和 stroke()完成填充和设置边框，通过 closePath()方法结束路径的绘制。

【例 9.2】（实例文件：ch09\9.2.html）

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="200" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#FFaa00";
cxt.beginPath();
cxt.arc(70,18,15,0,Math.PI*2,true);
cxt.closePath();
cxt.fill();
</script>
</body>
</html>
```

上面的 JavaScript 代码中，使用 beignPath 方法开启一个路径，然后绘制一个圆形，下面关闭这个路径并填充。

在 IE 9.0 中的浏览效果如图 9-2 所示，可以看到网页中，在矩形边框中显示了一个黄色的圆。



图 9-2　绘制椭圆

9.2.2　使用 moveTo 与 lineTo 绘制直线

在每个 canvas 实例对象中都拥有一个 path 对象，创建自定义图形的过程就是不断对 path

对象操作的过程。每当开始一次新的图形绘制任务，都需要先使用 beginPath()方法来重置 path 对象至初始状态，进而通过一系列对 moveTo/lineTo 等画线方法的调用，绘制期望的路径，其中 moveTo（x, y）方法设置绘图起始坐标，而 lineTo（x,y）等画线方法可以从当前起点绘制直线、圆弧以及曲线到目标位置。最后一步，也是可选的步骤，是调用 closePath()方法将自定义图形进行闭合，该方法将自动创建一条从当前坐标到起始坐标的直线。

绘制直线常用的方法是 moveTo 和 lineTo，其含义如下表所示。

方法或属性	功能
moveTo（x,y）	不绘制，只是将当前位置移动到新目标坐标（x,y），并作为线条开始点
lineTo（x,y）	绘制线条到指定的目标坐标（x,y），并且在两个坐标之间画一条直线。不管调用哪一个，都不会真正画出图形，因为还没有调用 stroke（绘制）和 fill（填充）函数。当前，只是在定义路径的位置，以便后面绘制时使用
strokeStyle	属性是指定线条的颜色
lineWidth	属性设置线条的粗细

【例 9.3】（实例文件：ch09\9.3.html）

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="200" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.beginPath();
cxt.strokeStyle="rgb(0,182,0)";
cxt.moveTo(10,10);
cxt.lineTo(150,50);
cxt.lineTo(10,50);
cxt.lineWidth=14;
cxt.stroke();
cxt.closePath();
</script>
</body>
</html>
```

上面代码中，使用 moveTo 方法定义一个坐标位置为（10,10），下面以此坐标位置为起点绘制了两个不同的直线，并使用 lineWidth 设置直线的宽带，使用 strokeStyle 设置了直线的颜色，使用 lineTo 设置了两个不同直线的结束位置。

在 IE 9.0 中的浏览效果如图 9-3 所示，可以看到网页中，绘制了两个直线，这两个直线在

某一点交叉。

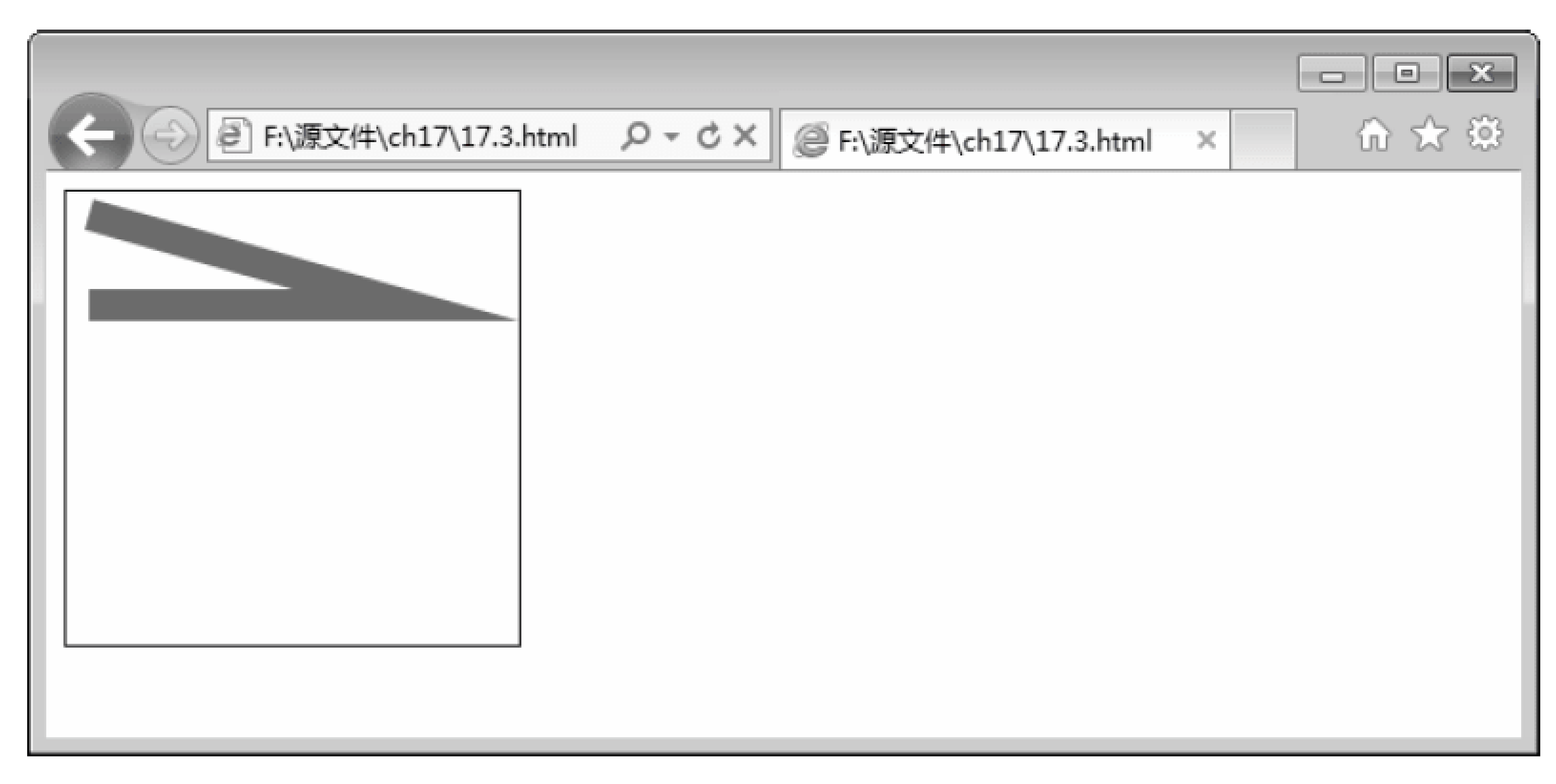


图 9-3　绘制直线

9.2.3　使用 bezierCurveTo 绘制贝济埃曲线

在数学的数值分析领域中，贝济埃曲线（Bézier 曲线）是计算机图形学中相当重要的参数曲线。更高维度的广泛化贝济埃曲线就称作贝济埃曲面，其中贝济埃三角是一种特殊的实例。

bezierCurveTo()表示为一个画布的当前子路径添加一条三次贝塞尔曲线。这条曲线的开始点是画布的当前点，而结束点是（x, y）。两条贝塞尔曲线控制点（cpX1, cpY1）和（cpX2, cpY2）定义了曲线的形状。当这个方法返回的时候，当前的位置为（x, y）。

方法 bezierCurveTo 具体格式如下所示：

```
bezierCurveTo(cpX1, cpY1, cpX2, cpY2, x, y)
```

其参数的含义如下表所示。

参数	描述
cpX1, cpY1	和曲线的开始点（当前位置）相关联的控制点的坐标
cpX2, cpY2	和曲线的结束点相关联的控制点的坐标
x, y	曲线的结束点的坐标

【例 9.4】（实例文件：ch09\9.4.html）

```
<!DOCTYPE html>
<html>
<head>
<title>贝济埃曲线</title>
<script>
    function draw(id)
    {
        var canvas=document.getElementById(id);
        if(canvas==null)
```

```
                return false;
                var context=canvas.getContext('2d');
                context.fillStyle="#eeeeff";
                context.fillRect(0,0,400,300);
                var n=0;
                var dx=150;
                var dy=150;
                var s=100;
                context.beginPath();
                context.globalCompositeOperation='and';
                context.fillStyle='rgb(100,255,100)';
                context.strokeStyle='rgb(0,0,100)';
                var x=Math.sin(0);
                var y=Math.cos(0);
                var dig=Math.PI/15*11;
                for(var i=0;i<30;i++)
                {
                    var x=Math.sin(i*dig);
                    var y=Math.cos(i*dig);
                    context.bezierCurveTo(dx+x*s,dy+y*s-100,dx+x*s+100,dy+y*s,dx
+x*s,dy+y*s);
                }
                context.closePath();
                context.fill();
                context.stroke();
            }
    </script>
    </head>
    <body onload="draw('canvas');">
    <h1>绘制元素</h1>
    <canvas id="canvas" width="400" height="300" />
    </body>
    </html>
```

在上述函数 draw 中，首先使用语句 fillRect（0,0,400,300）绘制了一个矩形，其大小和画布相同，其填充颜色为浅青色；然后定义变量，用于设定曲线的坐标位置，在 for 循环中使用 bezierCurveTo 绘制贝济埃曲线。

在 IE 9.0 中的浏览效果如图 9-4 所示，可以看到网页中显示了一个贝济埃曲线。

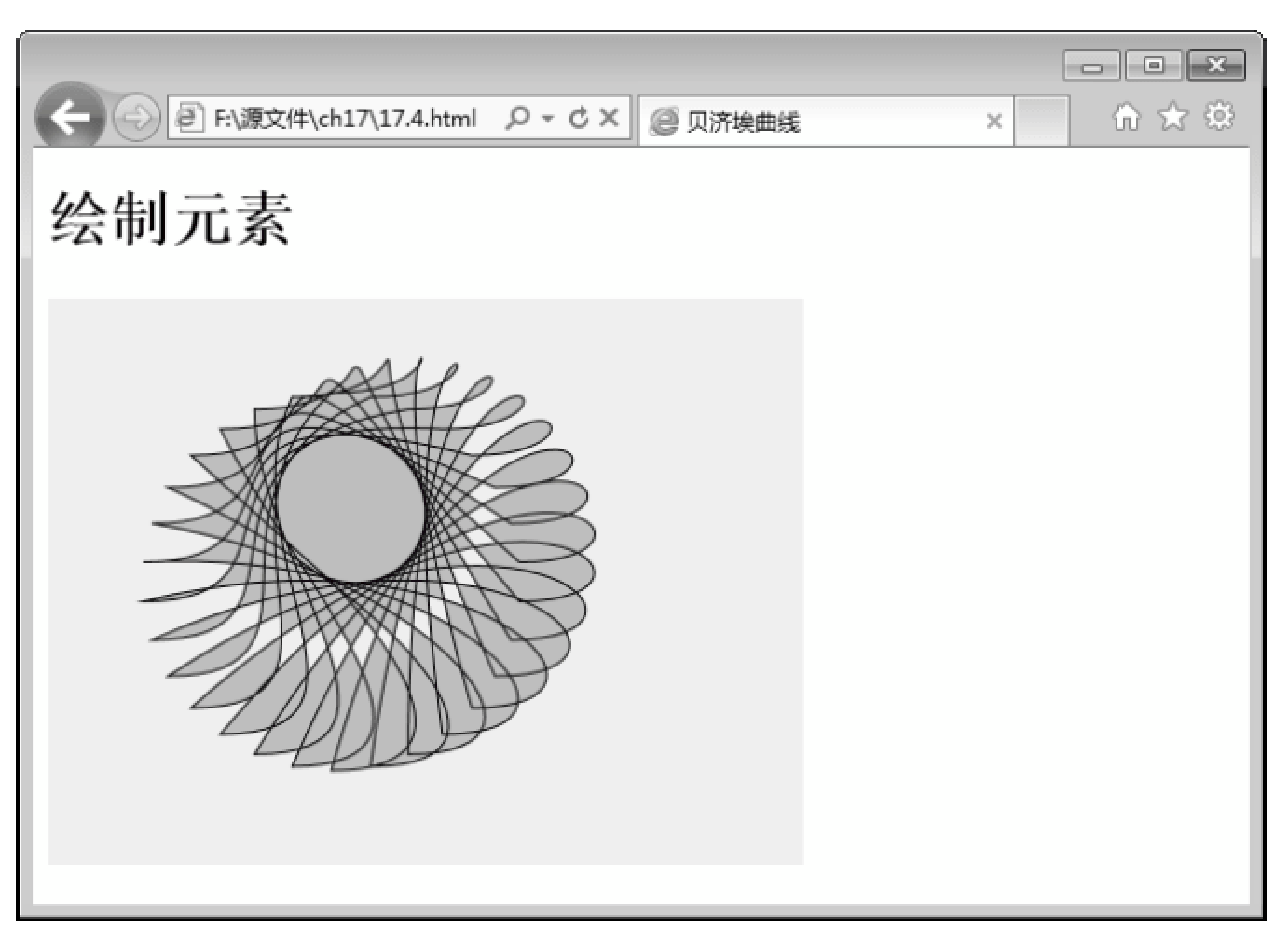


图 9-4 贝济埃曲线

9.3 绘制渐变图形

渐变是两种或更多颜色的平滑过度，是指在颜色集上使用逐步抽样算法将结果应用于描边样式和填充样式中。canvas 的绘图上下文支持两种类型的渐变：线性渐变和放射性渐变，其中放射性渐变也称为径向渐变。

9.3.1 绘制线性渐变

创建简单的渐变，非常容易，可能比使用 Photoshop 还要快，使用渐变需要三个步骤：

01 创建渐变对象，其代码如下：

```
var gradient=cxt.createLinearGradient(0,0,0,canvas.height);
```

02 为渐变对象设置颜色，指明过渡方式，其代码如下：

```
gradient.addColorStop(0,'#fff');
gradient.addColorStop(1,'#000');
```

03 在 context 上为填充样式或者描边样式设置渐变，其代码如下：

```
cxt.fillStyle=gradient;
```

要设置显示颜色，在渐变对象上使用 addColorStop 函数即可。除了可以变换成其他颜色外，还可以为颜色设置 alpha 值（例如透明），并且 alpha 值也是可以变化的。为了达到这样的效果，需要使用颜色值的另一种表示方法，例如内置 alpha 组件的 CSSrgba 函数。

绘制线性渐变，会使用到下面几个方法，如下表所示。

方法	功能
addColorStop	函数允许指定两个参数：颜色和偏移量。颜色参数是指开发人员希望在偏移位置描边或填充时所使用的颜色。偏移量是一个 0.0~1.0 之间的数值，代表沿着渐变线渐变的距离有多远。
createLinearGradient（x0,y0,x1,x1）	沿着直线从（x0,y0)至（x1,y1）绘制渐变

【例 9.5】（实例文件：ch09\9.5.html）

```
<!DOCTYPE html>
<html>
<head>
<title>线性渐变</title>
</head>
<body>
<h1>绘制线性渐变</h1>
<canvas id="canvas" width="400" height="300" style="border:1px solid red"/>
<script type="text/JavaScript">
var c=document.getElementById("canvas");
var cxt=c.getContext("2d");
var gradient=cxt.createLinearGradient(0,0,0,canvas.height);
gradient.addColorStop(0,'#fff');
gradient.addColorStop(1,'#000');
cxt.fillStyle=gradient;
cxt.fillRect(0,0,400,400);
</script>
</body>
</html>
```

上面的代码使用 2D 环境对象产生了一个线性渐变对象，渐变的起始点时(0,0)，渐变的结束点是(0,canvas.height)，下面使用 addColorStop 函数设置渐变颜色，最后将渐变填充到上下文环境的样式中。

在 IE 9.0 中的浏览效果如图 9-5 所示，可以看到网页中，创建了一个垂直方向上的渐变，从上到下颜色逐渐变深。



图 9-5　线性渐变

9.3.2　绘制径向渐变

除了线性渐变以外，HTML 5 Canvas API 还支持放射性渐变，所谓放射性渐变就是颜色会介于两个指定圆间的锥形区域平滑变化。放射性渐变和线性渐变使用的颜色终止点是一样的。如果要实现放射线渐变，即径向渐变，需要使用方法 createRadialGradient。

createRadialGradient（x0,y0,r0,x1,y1,r1）方法表示沿着两个圆之间的锥面绘制渐变。其中前三个参数代表开始的圆，圆心为(x0,y0)，半径为 r0。最后三个参数代表结束的圆，圆心为（x1,y1），半径为 r1。

【例 9.6】（实例文件：ch09\9.6.html）

```
<!DOCTYPE html>
<html>
<head>
<title>径向渐变</title>
</head>
<body>
<h1>绘制径向渐变</h1>
<canvas id="canvas" width="400" height="300" style="border:1px solid red"/>
<script type="text/JavaScript">
var c=document.getElementById("canvas");
var cxt=c.getContext("2d");
var gradient=cxt.createRadialGradient(canvas.width/2,canvas.height/2,0,canvas.width/2,canvas.height/2,150);
gradient.addColorStop(0,'#fff');
gradient.addColorStop(1,'#000');
cxt.fillStyle=gradient;
cxt.fillRect(0,0,400,400);
</script>
```

```
</body>
</html>
```

上面代码中，首先创建渐变对象 gradient，此处使用方法 createRadialGradient 创建了一个径向渐变，下面使用 addColorStop 添加颜色，最后将渐变填充到上下文环境中。

在 IE 9.0 中的浏览效果如图 9-6 所示，可以看到从圆的中心亮点开始，向外逐步发散，形成了一个径向渐变。

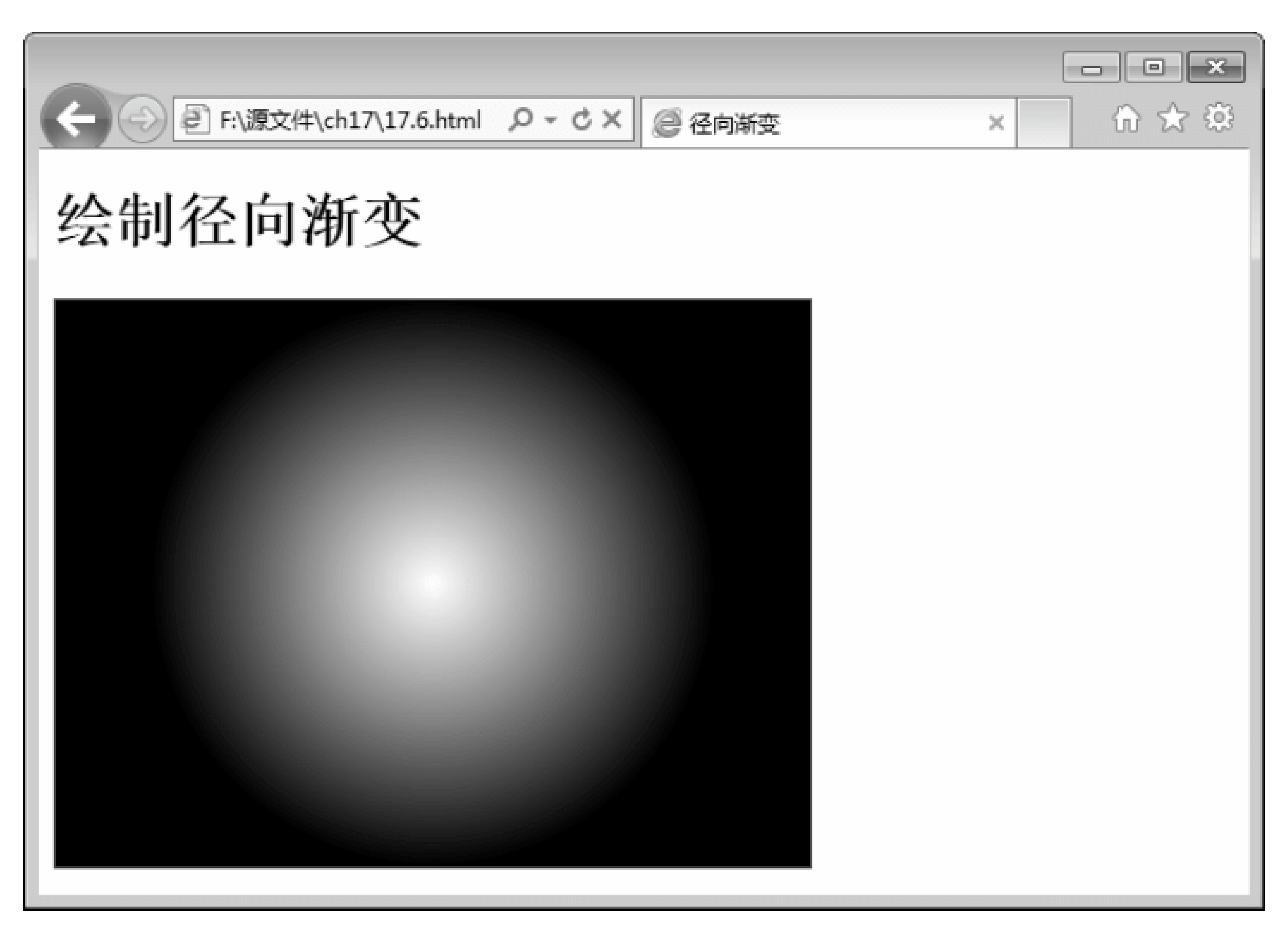


图 9-6　径向渐变

9.4　绘制变形图形

画布 canvas 不但可以使用 moveTo 这样的方法来移动画笔，绘制图形和线条，还可以使用变换来调整画笔下的画布。变换的方法包括：旋转、缩放、变形和平移等。

9.4.1　变换原点坐标

平移（translate），即将绘图区相对于当前画布的左上角进行平移，如果不进行变形，绘图区原点和画布原点是重叠的，绘图区相当于画图软件里的热区或当前层。如果进行变形，则坐标位置会移动到一个新位置。

如果要对图形实现平移，需要使用方法 translate(x,y)，该方法表示在平面上平移，即以原来的原点为参考，然后以偏移后的位置作为坐标原点，也就是说如果原来在(100,100)，translate(1,1)后，新的坐标原点在(101,101)而不是(1,1)。

【例 9.7】（实例文件：ch09\9.7.html）

```
<!DOCTYPE html>
<html>
<head>
<title>绘制坐标变换</title>
```

```
<script>
    function draw(id)
    {
        var canvas=document.getElementById(id);
        if(canvas==null)
        return false;
        var context=canvas.getContext('2d');
        context.fillStyle="#eeeeff";
        context.fillRect(0,0,400,300);
        context.translate(200,50);
        context.fillStyle='rgba(255,0,0,0.25)';
        for(var i=0;i<50;i++){
            context.translate(25,25);
            context.fillRect(0,0,100,50);
        }
    }
</script>
</head>
<body onload="draw('canvas');">
<h1>变换原点坐标</h1>
<canvas id="canvas" width="400" height="300" />
</body>
</html>
```

在 draw 函数中，使用 fillRect 方法绘制了一个矩形，在下面使用 translate 方法将其平移到了一个新位置，并从新位置开始，使用 for 循环，连续移动多次坐标原点，即多次绘制矩形。

在 IE 9.0 中的浏览效果如图 9-7 所示，可以看到网页中从坐标位置（200,50）开始绘制矩形，并每次以指定的平移距离绘制矩形。



图 9-7　变换坐标原点

9.4.2 图形缩放

对于变形图形来说，其中最常用的方式，就是对图形进行缩放，即以原来图形为参考，放大或者缩小图形，从而增加效果。

如果要实现图形缩放，需要使 scale（x,y）函数，该函数带有两个参数，分别代表在 x，y 两个方向上的值。每个参数在 canvas 显示图像的时候，向其传递在本方向轴上图像要放大（或者缩小）的量。如果 x 值为 2，就代表所绘制图像中全部元素都会变成两倍宽。如果 y 值为 0.5，绘制出来的图像全部元素都会变成之前的一半高。

【例 9.8】（实例文件：ch09\9.8.html）

```
<!DOCTYPE html>
<html>
<head>
<title>绘制图形缩放</title>
<script>
	function draw(id)
	{
		var canvas=document.getElementById(id);
		if(canvas==null)
		return false;
		var context=canvas.getContext('2d');
		context.fillStyle="#eeeeff";
		context.fillRect(0,0,400,300);
		context.translate(200,50);
		context.fillStyle='rgba(255,0,0,0.25)';
		for(var i=0;i<50;i++){
			context.scale(3,0.5);
			context.fillRect(0,0,100,50);
		}
	}
</script>
</head>
<body onload="draw('canvas');">
<h1>图形缩放</h1>
<canvas id="canvas" width="400" height="300" />
</body>
</html>
```

上面代码中，缩放操作是放在 for 循环中完成的，在此循环中，以原来图形为参考物，使其在 x 轴方向增加为三倍宽，y 轴方向上变为原来的一半。

在 IE 9.0 中的浏览效果如图 9-8 所示，可以看到网页中在一个指定方向绘制了多个矩形。



图 9-8　图形缩放

9.4.3　旋转图形

变换操作并不限于缩放和平移，还可以使用函数 context.rotate（angle）来旋转图像，甚至可以直接修改底层变换矩阵以完成一些高级操作，如剪裁图像的绘制路径。例如 context.rotate（1.57）表示旋转角度参数以弧度为单位。

rotate()方法默认地从左上端的(0,0)开始旋转，通过指定一个角度，改变画布坐标和 Web 浏览器中<canvas>元素像素之间的映射，使得任意后续绘图在画布中都显示为旋转的，但并没有旋转<canvas>元素本身。注意，这个角度是用弧度指定的。

【例 9.9】（实例文件：ch09\9.9.html）

```
<!DOCTYPE html>
<html>
<head>
<title>绘制旋转图像</title>
<script>
    function draw(id)
    {
        var canvas=document.getElementById(id);
        if(canvas==null)
        return false;
        var context=canvas.getContext('2d');
        context.fillStyle="#eeeeff";
        context.fillRect(0,0,400,300);
        context.translate(200,50);
        context.fillStyle='rgba(255,0,0,0.25)';
        for(var i=0;i<50;i++){
            context.rotate(Math.PI/10);
            context.fillRect(0,0,100,50);
```

```
            }
        }
</script>
</head>
<body onload="draw('canvas');">
<h1>旋转图形</h1>
<canvas id="canvas" width="400" height="300" />
</body>
</html>
```

上面代码中，使用 rotate 方法在 for 循环中对多个图形进行旋转，其旋转角度相同。

在 IE 9.0 中的浏览效果如图 9-9 所示，在显示页面上多个矩形以中心弧度为原点进行旋转。



图 9-9 旋转图形

9.5 图形组合

在前面介绍的知识里面，可以将一个图形画在另一个图形之上，大多数情况下，这样是不够的。例如说，这样会受制于图形的绘制顺序。不过，可以利用 globalCompositeOperation 属性来改变这些做法。不仅可以在已有图形后面再画新图形，还可以用来遮盖、清除（比 clearRect 方法强劲得多）某些区域。

其语法格式如下：

```
globalCompositeOperation = type
```

表示设置不同形状的组合类型，其中 type 表示方的图形是已存在的 canvas 内容，圆的图形是新的形状，其默认值为 source-over，表示在 canvas 内容上面画新的形状。

属性值 type 具有 12 个含义，其具体含义如下表所示。

属性值	说明
source-over（default）	这是默认设置，新图形会覆盖在原有内容之上
destination-over	会在原有内容之下绘制新图形
source-in	新图形会仅仅出现与原有内容重叠的部分。其他区域都变成透明的
destination-in	原有内容中与新图形重叠的部分会被保留，其他区域都变成透明的
source-out	结果是只有新图形中与原有内容不重叠的部分会被绘制出来
destination-out	原有内容中与新图形不重叠的部分会被保留
source-atop	新图形中与原有内容重叠的部分会被绘制，并覆盖于原有内容之上
destination-atop	原有内容中与新内容重叠的部分会被保留，并会在原有内容之下绘制新图形
lighter	两图形中重叠部分绘制两种颜色值相加的颜色
darker	两图形中重叠的部分绘制两种颜色值相减的颜色
xor	重叠的部分会变透明
copy	只有新图形会被保留，其他都被清除掉

【例 9.10】（实例文件：ch09\9.10.html）

```
<!DOCTYPE html>
<html>
<head>
<title>绘制图形组合</title>
<script>
function draw(id)
{
 var canvas=document.getElementById(id);
    if(canvas==null)
  return false;
    var context=canvas.getContext('2d');
    var oprtns=new Array(
          "source-atop",
            "source-in",
          "source-out",
        "source-over",
          "destination-atop",
        "destination-in",
          "destination-out",
          "destination-over",
            "lighter",
          "copy",
          "xor"
      );
     var i=10;
```

```
context.fillStyle="blue";
context.fillRect(10,10,60,60);
context.globalCompositeOperation=oprtns[i];
context.beginPath();
context.fillStyle="red";
context.arc(60,60,30,0,Math.PI*2,false);
context.fill();
}
</script>
</head>
<body onload="draw('canvas');">
<h1>图形组合</h1>
<canvas id="canvas" width="400" height="300" />
</body>
</html>
```

在上面的代码中，首先创建了一个 oprtns 数组，用于存储 type 的 12 个值，然后绘制了一个矩形，并使用 content 上下文对象设置了图形的组合方式，即采用了新图形显示，其他被清除的方式，最后使用 arc 绘制了一个圆。

在 IE 9.0 中的浏览效果如图 9-10 所示，在显示页面上绘制了一个矩形和圆，但矩形和圆接触的地方，以空白显示。



图 9-10　图形组合

9.6　绘制带阴影的图形

在画布 canvas 上绘制带有阴影效果的图形非常简单，只需要设置几个属性即可。这几个属性分别为 shadowOffsetX、shadowOffsetY、shadowBlur 和 shadowColor，其属性 shadowColor 表示阴影颜色，其值和 CSS 颜色值一致。shadowBlur 表示设置阴影模糊程度。此值越大，阴影越模糊。shadowOffsetX 和 shadowOffsetY 属性表示阴影的 x 和 y 偏移量，单位是像素。

【例 9.11】（实例文件：ch09\9.11.html）

```
<!DOCTYPE html>
```

```
<html>
 <head>
 <title>绘制阴影效果图形</title>
</head>
<body>
    <canvas id="my_canvas" width="200" height="200" style="border:1px solid #ff0000"></canvas>
    <script type="text/JavaScript">
        var elem = document.getElementById("my_canvas");
        if (elem && elem.getContext)  {
            var context = elem.getContext("2d");
            //shadowOffsetX 和 shadowOffsetY：阴影的 x 和 y 偏移量，单位是像素。
            context.shadowOffsetX = 15;
            context.shadowOffsetY = 15;
            //hadowBlur：设置阴影模糊程度。此值越大，阴影越模糊。其效果和 Photoshop 的高斯模糊滤镜相同。
            context.shadowBlur    = 10;
            //shadowColor：阴影颜色。其值和 CSS 颜色值一致。
            //context.shadowColor   = 'rgba(255, 0, 0, 0.5)';  或下面的十六进制的表示方法
            context.shadowColor = '#f00';
            context.fillStyle     = '#00f';
            context.fillRect(20, 20, 150, 100);
        }
    </script>
</body>
</html>
```

在 IE 9.0 中的浏览效果如图 9-11 所示，在显示页面上显示了一个蓝色矩形，其阴影为红色矩形。



图 9-11　带有阴影的图形

9.7 使用图像

画布 canvas 有一项功能就是可以引入图像，它可以用于图片合成或者制作背景等。而目前仅可以在图像中加入文字。只要是 Geck 支持的图像（如 PNG，GIF，JPEG 等）都可以引入到 canvas 中，而且其他的 canvas 元素也可以作为图像的来源。

9.7.1 绘制图像

要在画布 canvas 上绘制图像，需要先有一个图片。这个图片可以是已经存在的<img>元素，或者通过 JS 创建。无论采用哪种方式，都需要在绘制 canvas 之前，加载这张图片。浏览器通常会在页面脚本执行的同时异步加载图片。如果试图在图片未完全加载之前就将其呈现到 canvas 上，那么 canvas 将不会显示任何图片。

捕获和绘制图形完全是通过 drawImage 方法完成的，它可以接受不同的 HTML 参数，具体含义如下表所示。

方法	说明
drawIamge（image,dx,dy）	接受一个图片，并将之画到 canvas 中。给出的坐标(dx,dy)代表图片的左上角。例如，坐标(0,0)将把图片画到 canvas 的左上角。
drawIamge（image,dx,dy,dw,dh）	接受一个图片，将其缩放，宽度为 dw 和高度为 dh，然后把它画到 canvas 上的(dx,dy)位置
drawIamge（image,sx,sy,sw,sh,dx,dy,dw,dh）	接受一个图片，通过参数(sx,sy,sw,sh)指定图片裁剪的范围，缩放到(dw,dh)大小，最后把它画到 canvas 上的（dx,dy）位置

【例 9.12】（实例文件：ch09\9.12.html）

```
<!DOCTYPE html>
<html>
<head><title>绘制图像</title></head>
<body>
<canvas id="canvas" width="300" height="200" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
window.onload=function(){
    var ctx=document.getElementById("canvas").getContext("2d");
    var img=new Image();
    img.src="01.jpg";
    img.onload=function(){
        ctx.drawImage(img,0,0);
```

```
        }
    }
</script>
</body>
</html>
```

在上面代码中，使用窗口的 onload 加载事件，即页面被加载时执行函数。在函数中，创建上下文对象 ctx，并创建 Image 对象 img；下面使用 img 对象的属性 src 设置图片来源，最后使用 drawImage 画出当前的图像。

在 IE 9.0 中的浏览效果如图 9-12 所示，在页面上绘制了一个图像，并在画布中显示出来。



图 9-12　绘制图像

9.7.2　图像平铺

使用画布 canvas 绘制图像，有很多种用处，其中一个用处就是将绘制的图像作为背景图片使用。在做背景图片时，如果显示图片的区域大小不能直接设定，通常将图片以平铺的方式显示。

HTML 5 Canvas API 支持图片平铺，此时需要调用 createPattern 函数，即调用 createPattern 函数来替代之前的 drawImage 函数。函数 createPattern 的语法格式如下：

```
createPattern(image,type)
```

其中 image 表示要绘制的图像，type 表示平铺的类型，其具体含义如下表所示。

参数值	说明
no-repeat	不平铺
repeat-x	横方向平铺
repeat-y	纵方向平铺
repeat	全方向平铺

【例 9.13】（实例文件：ch09\9.13.html）

```
<!DOCTYPE html>
```

```
< html>
<head>
<title>绘制图像平铺</title>
</head>
<body onload="draw('canvas');">
<h1>图形平铺</h1>
<canvas id="canvas" width="400" height="300"></canvas>
<script>
    function draw(id){
        var canvas=document.getElementById(id);
        if(canvas==null){
            return false;
        }
        var context=canvas.getContext('2d');
        context.fillStyle="#eeeeff";
        context.fillRect(0,0,400,300);
        image=new Image();
        image.src="01.jpg";
        image.onload=function(){
            var ptrn=context.createPattern(image,'repeat');
            context.fillStyle=ptrn;
            context.fillRect(0,0,400,300);
        }
    }
</script>
</body>
</html>
```

上面代码中，使用 fillRect 创建了一个宽度为 400，高度为 300，左上角坐标位置为(0,0)的矩形，下面创建了一个 Image 对象，src 表示连接一个图像源，然后使用 createPattern 绘制一个图像，其方式是平铺，并将这个图像作为一个模式填充到矩形中。最后绘制这个矩形，此矩形大小完全覆盖原来的图形。

在 IE 9.0 中的浏览效果如图 9-13 所示，在显示页面上绘制了一个图像，其图像以平铺的方式充满整个矩形。



图 9-13　图像平铺

9.7.3　图像裁剪

在处理图像时经常会遇到裁剪这种需求，即在画布上裁剪出一块区域，这块区域是在裁剪动作 clip 之前，由绘图路径设定的，可以是方形、圆形、五星形和其他任何可以绘制的轮廓形状。所以，裁剪路径其实就是绘图路径，只不过这个路径不是拿来绘图的，而是设定显示区域和遮挡区域的一个分界线。

完成对图像的裁剪，可能要用到 clip 方法。clip 方法表示给 canvas 设置一个剪辑区域，在调用 clip 方法之后的代码只对这个设定的剪辑区域有效，不会影响其他地方，这个方法在要进行局部更新时很有用。默认情况下，剪辑区域是一个左上角在（0,0），宽和高分别等于 canvas 元素的宽和高的矩形。

【例 9.14】（实例文件：ch09\9.14.html）

```
<!DOCTYPE html>
< html>
<head>
<title>绘制图像裁剪</title>
<script type="text/JavaScript" src="script.js"></script>
</head>
<body onload="draw('canvas');">
<h1>图像裁剪实例</h1>
<canvas id="canvas" width="400" height="300"></canvas>
<script>
    function draw(id){
        var canvas=document.getElementById(id);
```

```
            if(canvas==null){
                return false;
            }
            var context=canvas.getContext('2d');
            var gr=context.createLinearGradient(0,400,300,0);
            gr.addColorStop(0,'rgb(255,255,0)');
            gr.addColorStop(1,'rgb(0,255,255)');
            context.fillStyle=gr;
            context.fillRect(0,0,400,300);
            image=new Image();
            image.onload=function(){
                drawImg(context,image);
            };
            image.src="01.jpg";
        }
        function drawImg(context,image){
            create8StarClip(context);
            context.drawImage(image,-50,-150,300,300);
        }
        function create8StarClip(context){
            var n=0;
            var dx=100;
            var dy=0;
            var s=150;
            context.beginPath();
            context.translate(100,150);
            var x=Math.sin(0);
            var y=Math.cos(0);
            var dig=Math.PI/5*4;
            for(var i=0;i<8;i++){
                var x=Math.sin(i*dig);
                var y=Math.cos(i*dig);
                context.lineTo(dx+x*s,dy+y*s);
            }
            context.clip();
        }
    </script>
    </body>
    </html>
```

上面代码中，创建了三个 JavaScript 函数，其中 create8StarClip 函数完成了多边的图形创建，并以此图形作为裁剪的依据。drawImg 函数表示绘制一个图形，其图形带有裁剪区域。draw

函数完成对画布对象的获取，并定义线性渐变，然后创建 Image 对象。

在 IE 9.0 中浏览效果如图 9-14 所示，在显示页面上绘制如图所示图形，将图像作为该图形的背景显示，从而实现对图像的裁剪。



图 9-14　图像裁剪

9.7.4　像素处理

在计算机屏幕上可以看到色彩斑斓的图像，其实这些图像都是由一个个像素点组成的。一个像素对应着内存中的一组连续的二进制位，由于是二进制位，每个位上的取值当然只能是 0 或者 1 了，这样，这组连续的二进制位就可以由 0 和 1 排列组合出很多种情况，而每一种排列组合就决定了这个像素的一种颜色。因此，每个像素点由 4 个字节组成。

这 4 个字节代表含义分别是，第一个字节决定像素的红色值；第二个字节决定像素的绿色值；第三个字节决定像素的蓝色值；第四个字节决定像素的透明度值。

在画布中，可以使用 ImageData 对象来保存图像像素值，它有 width、height 和 data 三个属性，其中 data 属性就是一个连续数组，图像的所有像素值其实是保存在 data 里面的。

data 属性保存像素值的方法：

```
imageData.data[index*4 +0]
imageData.data[index*4 +1]
imageData.data[index*4 +2]
imageData.data[index*4 +3]
```

上面取出了 data 数组中连续相邻的 4 个值，这 4 个值分别代表了图像中第 index+1 个像素的红色、绿色、蓝色和透明度值的大小。需要注意的是 index 从 0 开始，图像中总共有 width * height 个像素，数组中总共保存了 width * height * 4 个数值。

画布对象有三个方法用来创建、读取和设置 ImageData 对象，如下表所示。

方法	说明
createImageData（width, height）	在内存中创建一个指定大小的 ImageData 对象（即像素数组），对象中的像素点都是黑色透明的，即 rgba(0,0,0,0)
getImageData（x, y, width, height）	返回一个 ImageData 对象，这个 IamgeData 对象中包含了指定区域的像素数组
putImageData（data, x, y）	将 ImageData 对象绘制到屏幕的指定区域上

【例 9.15】（实例文件：ch09\9.15.html）

```
<!DOCTYPE html>
< html>
<head>
<title>图像像素处理</title>
<script type="text/JavaScript" src="script.js"></script>
</head>
<body onload="draw('canvas');">
<h1>像素处理示例</h1>
<canvas id="canvas" width="400" height="300"></canvas>
<script>
    function draw(id){
        var canvas=document.getElementById(id);
        if(canvas==null){
            return false;
        }
        var context=canvas.getContext('2d');
        image=new Image();
        image.src="01.jpg";
        image.onload=function(){
            context.drawImage(image,0,0);
            var imagedata=context.getImageData(0,0,image.width,image.height);
            for(var i=0,n=imagedata.data.length;i<n;i+=4){
                imagedata.data[i+0]=255-imagedata.data[i+0];
                imagedata.data[i+1]=255-imagedata.data[i+2];
                imagedata.data[i+2]=255-imagedata.data[i+1];
            }
            context.putImageData(imagedata,0,0);
        };
    }
</script>
</body>
</html>
```

在上面代码中，使用 getImageData 方法获取一个 ImageData 对象，并包含相关的像素数组。在 for 循环中，对像素值重新赋值，最后使用 putImageData 将处理过的图像在画布上绘制出来。

在 IE 9.0 中的浏览效果如图 9-15 所示，在显示页面上显示了一个图像，其图像明显经过像素处理，显示没有原来清晰。



图 9-15　像素处理

9.8　绘制文字

在画布中绘制字符串（文字）的方式，与操作其他路径对象的方式相同，可以描绘文本轮廓和填充文本内部，同时，所有能够应用于其他图形的变换和样式都能用于文本。

文本绘制功能由两个函数组成，如下表所示。

方法	说明
fillText（text,x,y,maxwidth）	绘制带 fillStyle 填充的文字，文本参数以及用于指定文本位置的坐标参数。maxwidth 是可选参数，用于限制字体大小，它会将文本字体强制收缩到指定尺寸
trokeText（text,x,y,maxwidth）	绘制只有 strokeStyle 边框的文字，其参数含义和方法与 fillText 相同
measureText	该函数会返回一个度量对象，其包含了在当前 context 环境下指定文本的实际显示宽度

为了保证文本在各浏览器下都能正常显示，在绘制上下文里有以下字体属性：

- font 可以是 CSS 字体规则中的任何值。包括字体样式、字体变种、字体大小与粗细、行高和字体名称。
- textAlign 控制文本的对齐方式。它类似于（但不完全相同）CSS 中的 text-align。可能的取值为 start、end、left、right 和 center。
- textBaseline 控制文本相对于起点的位置。可以取值有 top、hanging、middle、alphabetic、ideographic 和 bottom。对于简单的英文字母，可以放心的使用 top、middle 或 bottom 作为文本基线。

【例 9.16】（实例文件：ch09\9.16.html）

```
<!DOCTYPE html>
<html>
 <head>
  <title>Canvas</title>
 </head>
 <body>
     <canvas id="my_canvas" width="200" height="200" style="border:1px solid #ff0000"></canvas>
     <script type="text/JavaScript">
          var elem = document.getElementById("my_canvas");
          if (elem && elem.getContext)   {
               var context = elem.getContext("2d");
               context.fillStyle      = '#00f';
               //font：文字字体，同 CSSfont-family 属性
               context.font = 'italic   30px 微软雅黑';      //斜体 30 像素 微软雅黑字体
               //textAlign：文字水平对齐方式。可取属性值: start, end, left,right, center。默认值:start.
               context.textAlign = 'left';
               //文字竖直对齐方式。可取属性值：top, hanging, middle,alphabetic, ideographic, bottom。
默认值：alphabetic
               context.textBaseline = 'top';
               //要输出的文字内容，文字位置坐标，第四个参数为可选选项——最大宽度。如果需要的
话，浏览器会缩减文字以让它适应指定宽度
               context.fillText   ('祖国生日快乐!', 0, 0,50);      //有填充
               context.font              = 'bold 30px sans-serif';
               context.strokeText('祖国生日快乐!', 0, 50,100);   //只有文字边框
          }
     </script>
 </body>
 </html>
```

在 IE 9.0 中的浏览效果如图 9-16 所示，在页面上显示画布边框，画布中显示了两个不同的字符串，第一个字符串以斜体显示，颜色为蓝色。第二个字符串字体颜色为黑色，加粗显示。

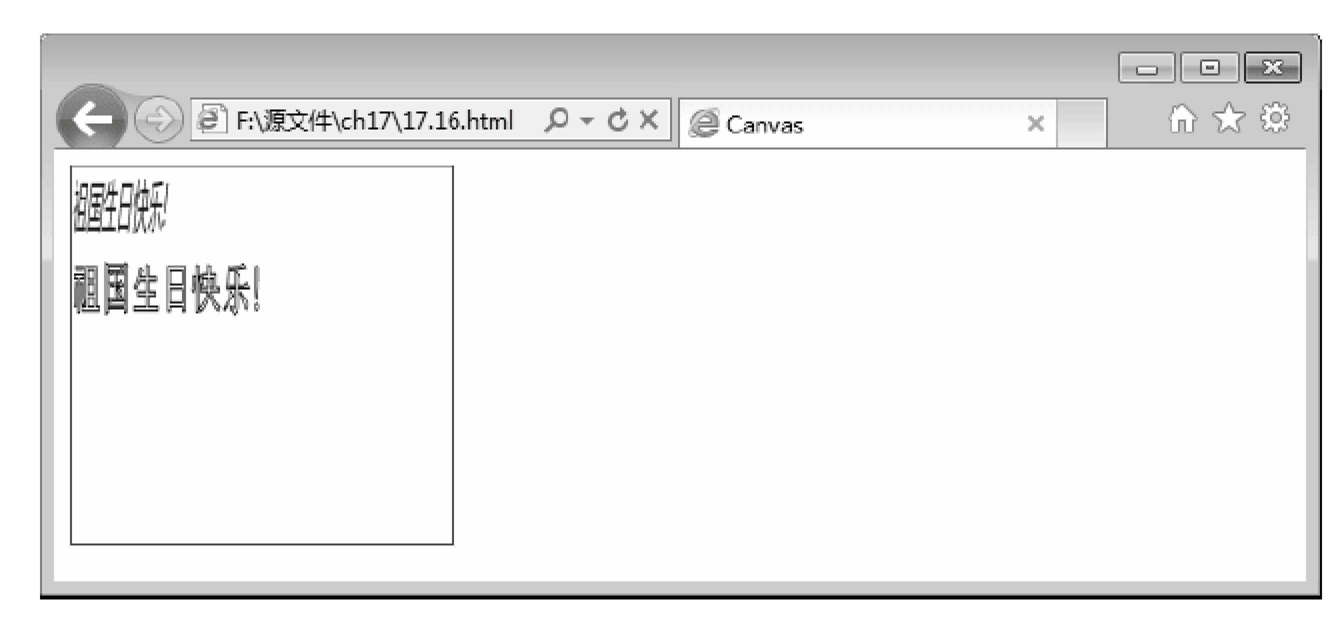


图 9-16　绘制文字

9.9　图形的保存与恢复

在画布对象绘制图形或图像时，可以将这些图形或者图形的状态进行改变，即永久保存图形或图像。

9.9.1　保存与恢复状态

在画布对象中，有两个方法管理绘制状态的当前栈，save 方法把当前状态压入栈中，而 restore 方法从栈顶弹出状态，绘制状态不会覆盖对画布所做的每件事情，其中 save 方法用来保存 canvas 的状态。调用完 save 方法之后，可以调用 canvas 进行平移、放缩、旋转、错切和裁剪等操作。restore 方法用来恢复 canvas 之前保存的状态，防止调用 save 方法后对 canvas 执行的操作对后续的绘制有影响。save 方法和 restore 方法要配对使用（restore 方法可以比 save 方法少，但不能多），如果 restore 方法调用次数比 save 方法多，会引发 Error。

【例 9.17】（实例文件：ch09\9.17.html）

```
<!DOCTYPE html>
<html>
<head><title>保存与恢复</title></head>
<body>
<canvas id="myCanvas" width="500" height="400" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle = "rgb(0,0,255)";
ctx.save();
ctx.fillRect(50,50,100,100);
ctx.fillStyle = "rgb(255,0,0)";
ctx.save();
ctx.fillRect(200,50,100,100);
ctx.restore()
ctx.fillRect(350,50,100,100);
ctx.restore();
ctx.fillRect(50, 200, 100, 100);
</script>
</body>
</html>
```

在上面代码中，绘制了 4 个矩形，在第一个矩形绘制之前，定义了当前矩形的显示颜色，并将此样式加入到栈中，然后创建矩形。第二个矩形绘制之前，重新定义矩形显示颜色，并使用 save 方法将此样式压入到栈中，然后创建了一个矩形。在第三个矩形绘制之前，使用 restore

方法恢复当前显示颜色，即调用栈中的最上层颜色，绘制矩形。第四个矩形绘制之前，继续使用 restore 方法，调用最后一个栈中元素定义矩形颜色。

在 IE 9.0 中的浏览效果如图 9-17 所示，在页面上绘制了 4 个矩形，第一个和第四个矩形显示为蓝色，第二个和第三个矩形显示为红色。

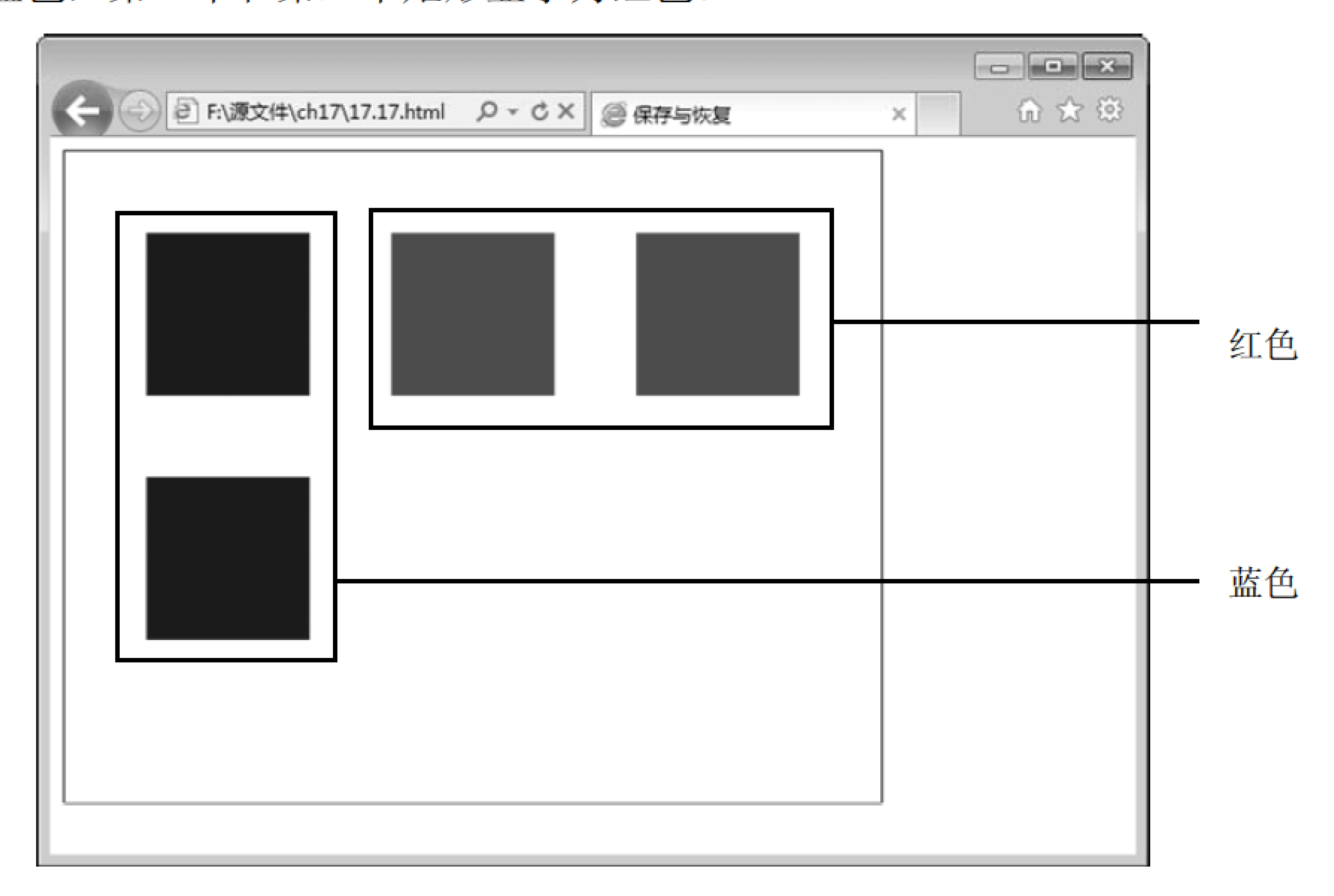


图 9-17　恢复和保存

9.9.2　保存文件

当绘制出漂亮的图形时，有时需要保存这些劳动成果。这时可以将画布元素（而不是 2D 环境）的当前状态导出到 URL 数据。导出很简单，可以利用 toDataURL 方法完成，它可以调用不同的图片格式。目前 PNG 格式才是定义的规范格式，其他浏览器还支持其他的格式。

目前 Firefox 和 Opera 浏览器只支持 PNG 格式，Safari 支持 GIF、PNG 和 JPG 格式。大多数浏览器支持读取 base64 编码内容，例如一幅图像。URL 的格式如下。

```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAfQAAAH0CAYAAADL1t
```

它以一个 data 开始，然后是 mine 类型，之后是编码和 base64，最后是原始数据。这些原始数据就是画布元素所要导出的内容，并且浏览器能够将数据编码为真正的资源。

【例 9.18】（实例文件：ch09\9.9.html）

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="500" height="500" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
<script type="text/JavaScript">
```

```
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle='rgb(0,0,255)';
cxt.fillRect(0,0,cxt.canvas.width,cxt.canvas.height);
cxt.fillStyle="rgb(0,255,0)";
cxt.fillRect(10,20,50,50);
window.location=cxt.canvas.toDataURL(image/png ');
</script>
</body>
</html>
```

在上面代码中，使用 canvas.toDataURL 语句将当前绘制的图像保存到 URL 数据中。

在 IE 9.0 中浏览效果如图 9-18 所示，在显示页面中无任何数据显示，并且提示无法显示该页面。此时需要注意的是地址栏中的 URL 信息。



图 9-18　保存图形

9.9.3　绘制图形综合应用

绘制商标是 canvas 画布的用途之一，可以绘制 adidas 和 Nike 商标。Nike 的商标比 adidas 的商标复杂的多，adidas 都是由直线组成的，而 Nike 的多了曲线。实现本实例步骤如下所示。

01 分析需求。要绘制两条曲线，需要找到曲线的参考点（参考点决定了曲线的曲率），这需要慢慢地移动，然后再看效果，要反复试验。quadraticCurveTo（30,79,99,78）函数有两组坐标，第一组坐标为控制点，决定曲线的曲率，第二组坐标为终点。

02 构建 HTML，实现 canvas 画布。在 IE 9.0 中浏览效果如图 9-19 所示，此时只显示一个画布边框，其内容还没有绘制。其代码如下:

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>绘制商标</title>
</head>
<body>
<canvas id="adidas" width="375px" height="132px" style="border:1px solid #000;"></canvas>
</body>
</html>
```



图 9-19　定义画布边框

03 用 JavaScript 实现基本图形：在 IE 9.0 中浏览效果如图 9-20 所示，显示了一个商标图案，其代码如下：

```
<script>
function drawAdidas(){
    //取得 convas 元素及其绘图上下文
    var canvas=document.getElementById('adidas');
    var context=canvas.getContext('2d');
    //保存当前绘图状态
    context.save();
    //开始绘制打勾的轮廓
    context.beginPath();
    context.moveTo(53,0);
    //绘制上半部分曲线，第一组坐标为控制点，决定曲线的曲率，第二组坐标为终点
    context.quadraticCurveTo(30,79,99,78);
    context.lineTo(371,2);
    context.lineTo(74,134);
    context.quadraticCurveTo(-55,124,53,0);
    //用红色填充
    context.fillStyle="#da251c";
    context.fill();
    //用 3 像素深红线条描边
    context.lineWidth=3;
    //连接处平滑
    context.lineJoin='round';
```

```
        context.strokeStyle="#d40000";
        context.stroke();
        //恢复原有绘图状态
        context.restore();
    }
    window.addEventListener("load",drawAdidas,true);
    </script>
```

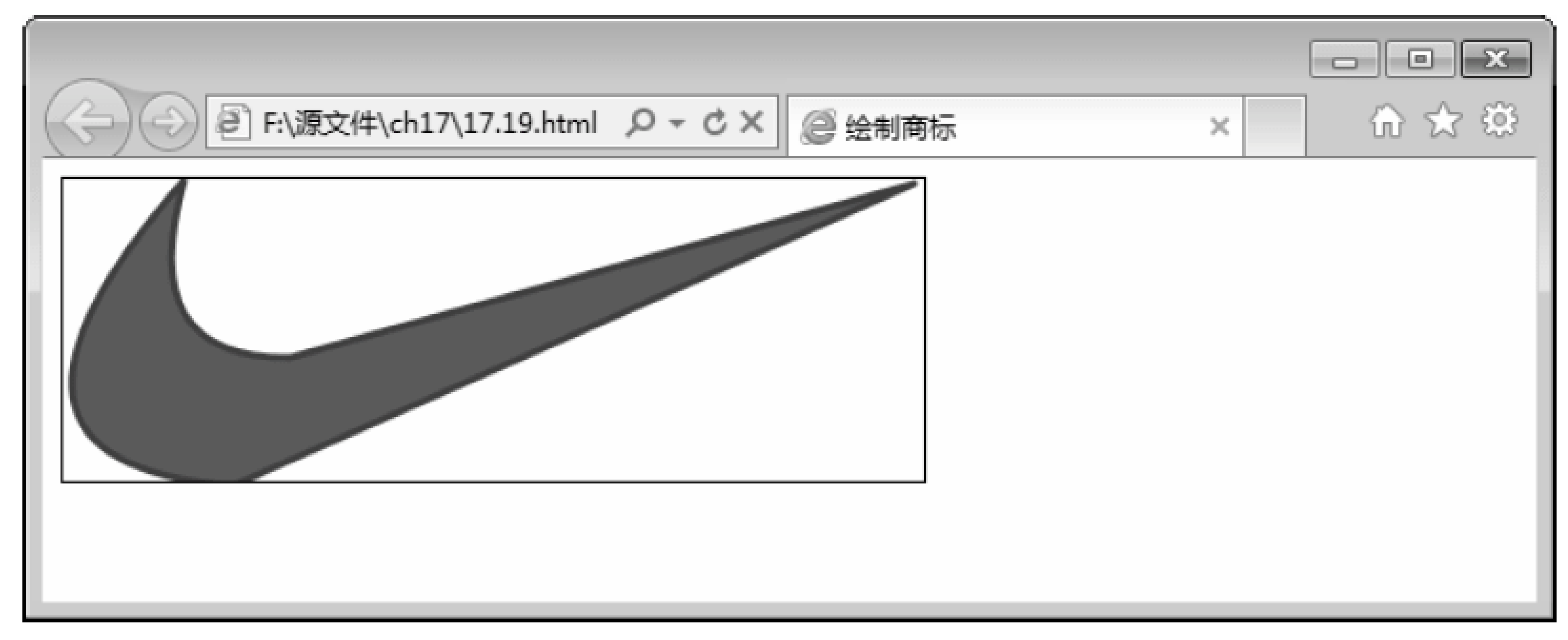


图 9-20　绘制的商标

9.10　综合实例——绘制火柴棒人物

漫画中最常见的一种图形，就是火柴棒人，通过简单的几个笔画，就可以绘制一个传神的动漫人物。使用 canvas 和 JavaScript 同样可以绘制一个火柴棒人物。具体步骤如下所示。

01 分析需求。一个火柴棒人，由以下几个部分组成：脸部，身躯。脸部是一个圆形，其中包括眼睛和嘴；身躯由几条直线组成，包括手和腿等。实际上此案例就是绘制圆形、弧度和直线的组合。实例完成后，效果如图 9-21 所示。

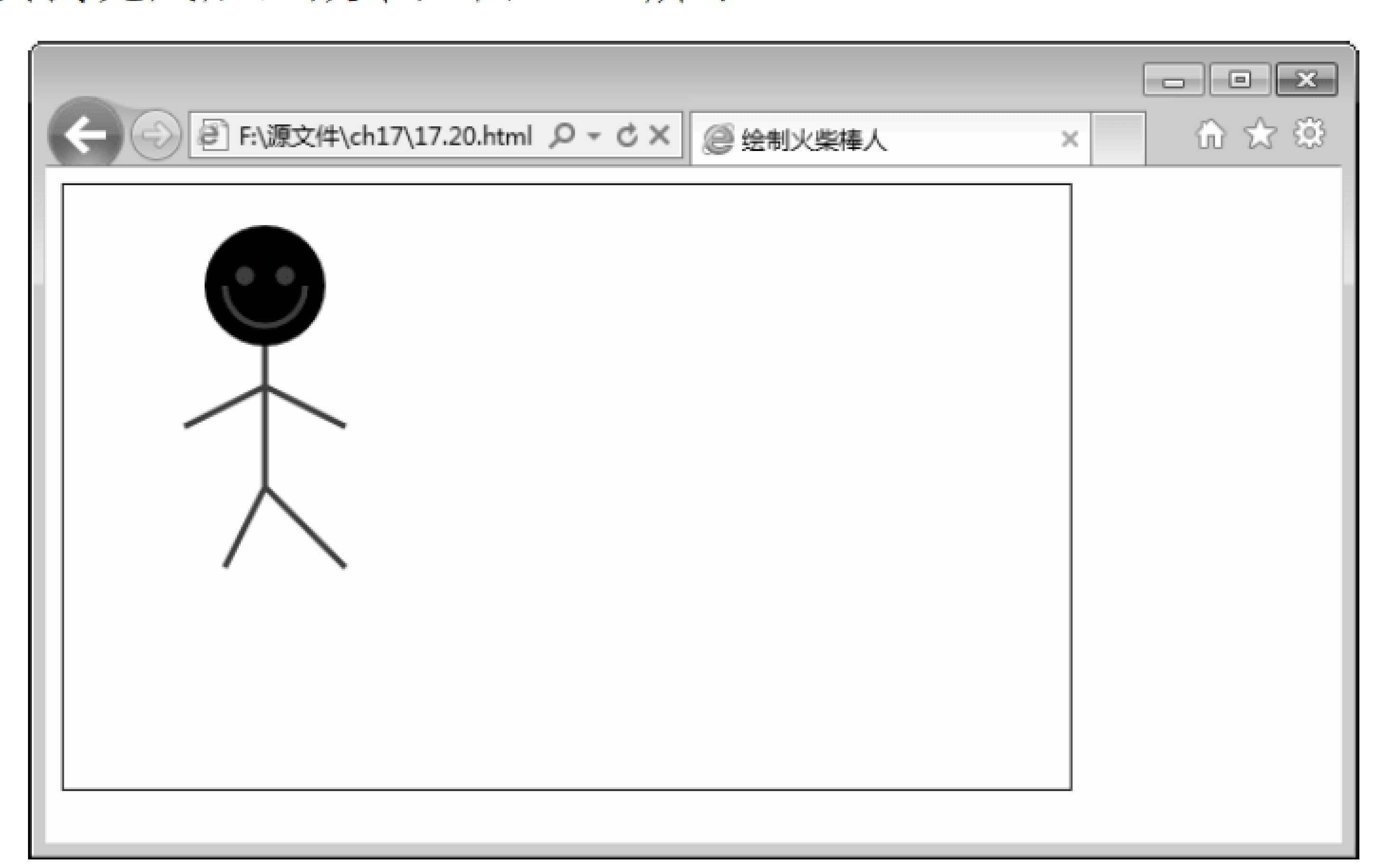


图 9-21　火柴棒人物

02 在 HTML 页面实现定义画布 canvas。在 IE 9.0 中浏览效果如图 9-22 所示，页面中显示了一个画布边框。其代码如下：

```
<!DOCTYPE html>
<html>
<title>绘制火柴棒人</title>
<body>
<canvas id="myCanvas" width="500" height="300" style="border:1px solid blue">
Your browser does not support the canvas element.
</canvas>
</body>
</html>
```



图 9-22　定义画布边框

03 实现头部轮廓绘制，代码如下：

```
<script type="text/JavaScript">
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
cxt.beginPath();
cxt.arc(100,50,30,0,Math.PI*2,true);
cxt.fill();
</script>
```

04 执行程序后，将会产生一个实心的、填充的头部，即圆形。在 arc 函数中，x 和 y 的坐标为(100,50)，半径为 30 像素，另两个参数为弧度的开始和结束，第六个参数表示绘制弧形的方向，即顺时针和逆时针方向。在 IE 9.0 中浏览效果如图 9-23 所示，页面中显示了一个实心圆，其颜色为黑色。

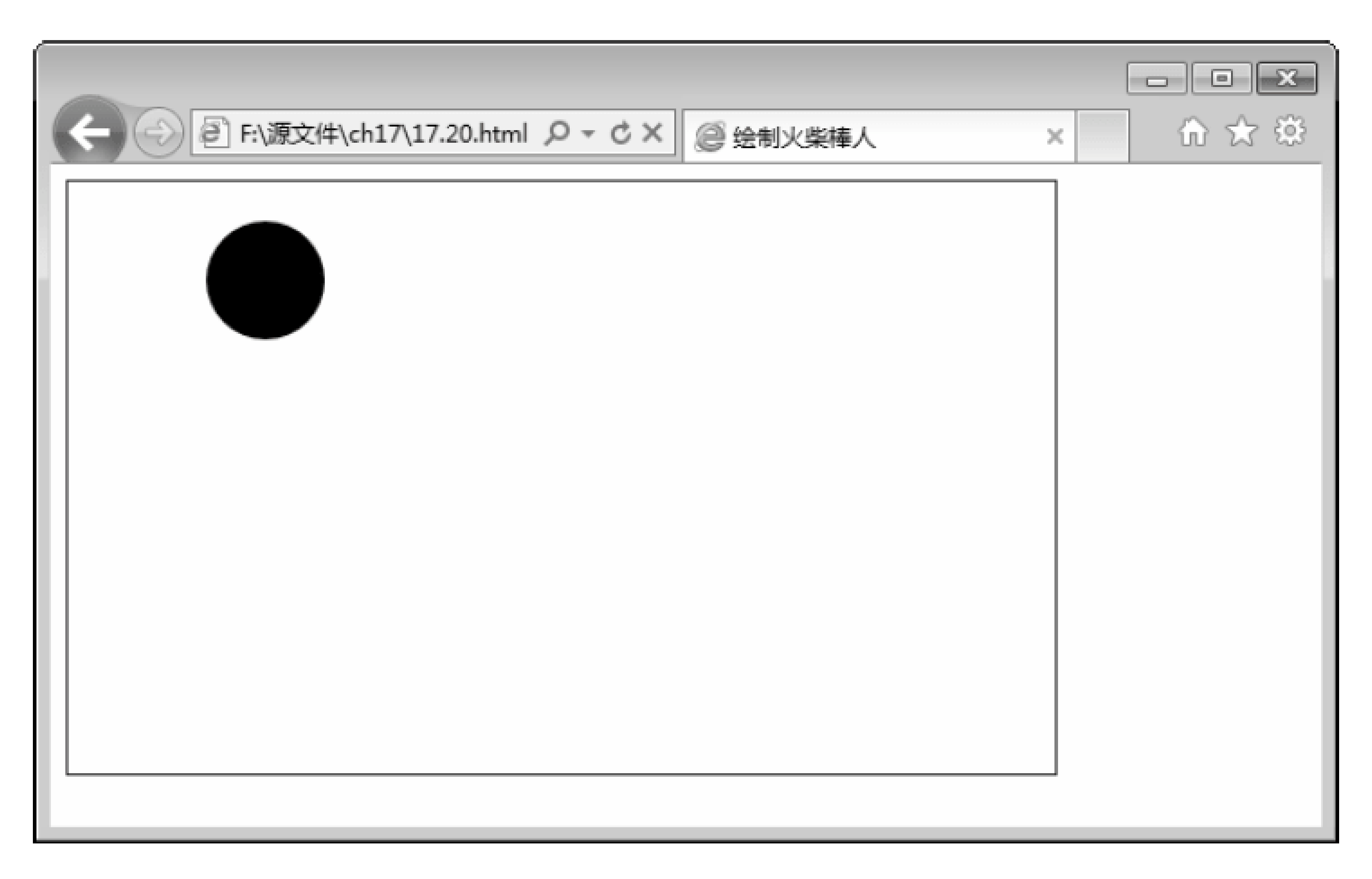


图 9-23　绘制头部轮廓

05 用 JavaScript 绘制笑脸，代码如下：

```
cxt.beginPath();
cxt.strokeStyle='#c00';
cxt.lineWidth=3;
cxt.arc(100,50,20,0,Math.PI,false);
cxt.stroke();
```

06 此处使用 beginPath 方法，表示重新绘制，并设定线条宽度，然后绘制弧形，从嘴角开始。在 IE 9.0 中浏览效果如图 9-24 所示，页面上显示了一个漂亮的半圆式的笑脸。



图 9-24　绘制笑脸

07 用 JavaScript 绘制眼睛：首先填充弧线，创建了一个实体样式的眼睛，arc 绘制左眼，然后使用 moveto 绘制右眼。在 IE 9.0 中浏览效果如图 9-25 所示，页面显示了一双眼睛。其代码如下：

```
cxt.beginPath();
cxt.fillStyle="#c00";
```

```
cxt.arc(90,45,3,0,Math.PI*2,true);
cxt.fill();
cxt.moveTo(113,45);
cxt.arc(110,45,3,0,Math.PI*2,true);
cxt.fill();
cxt.stroke();
```

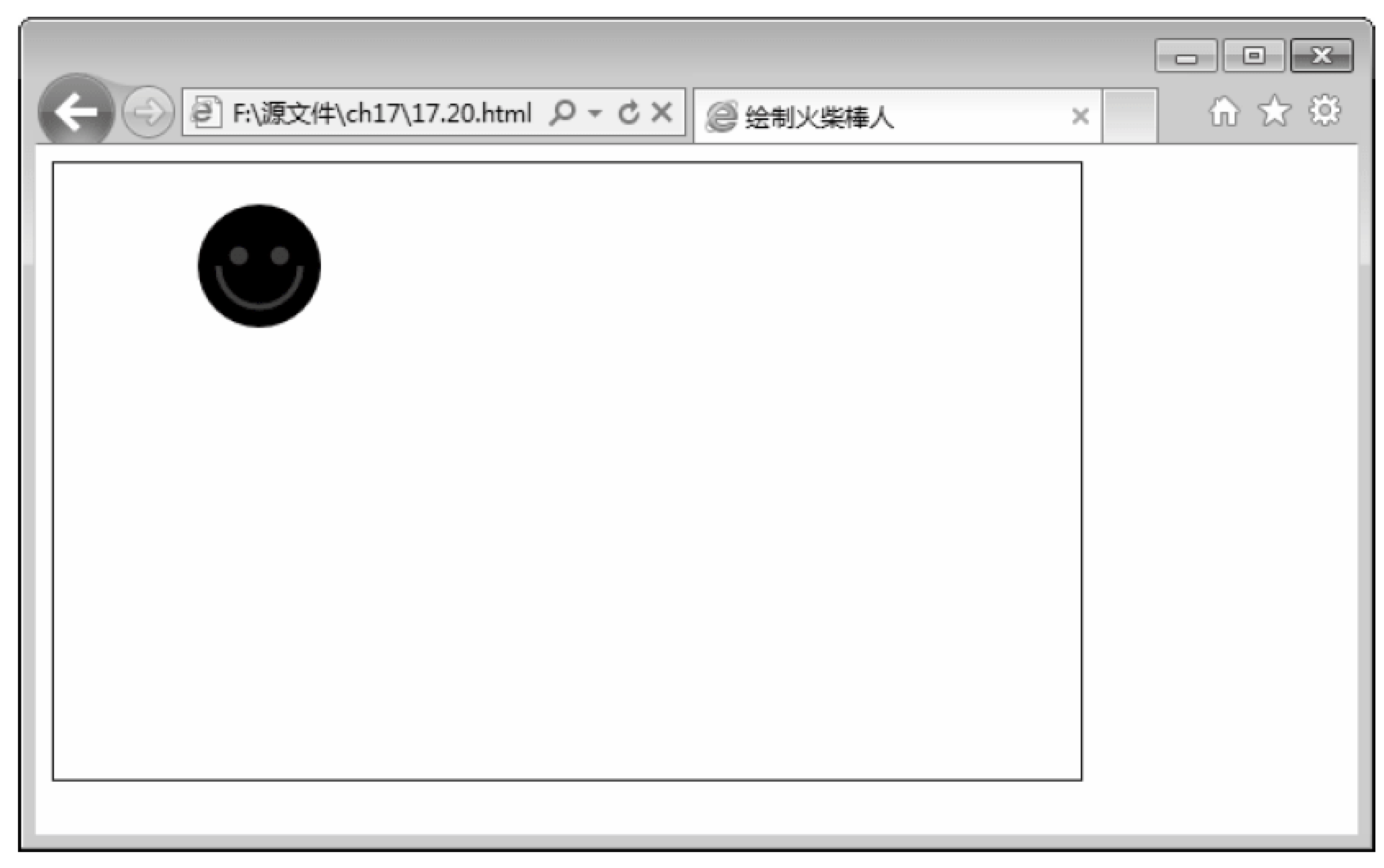


图 9-25　绘制眼睛

08 绘制身躯，代码如下：

```
cxt.moveTo(100,80);
cxt.lineTo(100,150);
cxt.moveTo(100,100),
cxt.lineTo(60,120);
cxt.moveTo(100,100);
cxt.lineTo(140,120);
cxt.moveTo(100,150);
cxt.lineTo(80,190);
cxt.moveTo(100,150);
cxt.lineTo(140,190);
cxt.stroke();
```

上面代码以 moveTo 作为开始坐标，以 lineTo 为终点，绘制不同的直线，这些直线的坐标位置需要在不同地方汇集，两只手在坐标(100,100)以上交叉，两只脚在坐标(100,150)处交叉。

在 IE 9.0 中浏览效果如图 9-21 所示，页面显示了一个火柴棒人物，相比较图 9-24，多了一个身躯。

9.11　问题解答

1．定义 canvas 宽度和高度时，是否可以在 CSS 属性中定义？

添加 canvas 标签时，会在 canvas 属性里填写要初始化的 canvas 的高度和宽度：

```
<canvas width="500" height="400">Not Supported!</canvas>
```

如果把高度和宽度写在 CSS 里面，结果发现在绘图的时候坐标获取出现差异，canvas.width 和 canvas.height 分别是 300 和 150，和预期的不一样。这是因为 canvas 要求这两个属性必须 canvas 和标记一起出现。

2．画布中 Stroke 和 Fill 二者的区别是什么？

HTML 5 中将图形分为两大类：第一类称作 Stroke，就是轮廓、勾勒或者线条，总之，图形是由线条组成的；第二类称作 Fill，就是填充区域。上下文对象中有两个绘制矩形的方法，可以让程序员更好地理解这两大类图形的区别：strokeRect 和 fillRect。

第 10 章　HTML 5 中的音频和视频

目前，在网页上没有关于音频和视频的标准，多数音频和视频都是通过插件来播放的。为此，HTML 5 新增了音频和视频的标签。本章将讲述音频和视频的基本概念、常用属性、解码器和浏览器的支持情况。

10.1　audio 标签

目前，大多数是通过插件来播放音频文件的，例如常见的播放插件为 Flash，这就是为什么用户在用浏览器播放音乐时，常常需要安装 Flash 插件的原因。但是，并不是所有的浏览器都拥有同样的插件。

为此，和 HTML 4 相比，HTML 5 新增了 audio 标签，规定了一种包含音频的标准方法。

10.1.1　audio 标签概述

audio 标签是定义播放声音文件或者音频流的标准。支持三种音频格式，分别为 Ogg、MP3 和 Wav。

如果需要在 HTML 5 网页中播放音频，输入的基本格式如下：

```
<audio src="song.mp3" controls="controls">
</audio>
```



其中 src 属性是规定要播放的音频地址，controls 属性供添加播放、暂停和音量控件用的。

另外，在<audio>与</audio>之间插入的内容是供不支持 audio 元素的浏览器显示的。

【例 10.1】（实例文件：ch10\10.1.html）

```
<!DOCTYPE html>
<html>
<head>
<title>audio</title>
<head>
<body >
  <audio src="song.mp3" controls="controls">
您的浏览器不支持 audio 标签！
```

```
</audio>
</body>
</html>
```

如果用户的浏览器是 IE 9.0 以前的版本，浏览效果如图 9-1 所示，可见 IE 9.0 以前的版本浏览器不支持 audio 标签。

在 Firefox 8.0 中浏览效果如图 10-2 所示，可以看到加载的音频控制条和听到加载的音频文件。



图 10-1　不支持 audio 标签的效果



图 10-2　支持 audio 标签的效果

10.1.2　audio 标签的属性

Audio 标签的常见属性和含义如下表所示。

属性	值	描述
autoplay	Autoplay（自动播放）	如果出现该属性，则音频在就绪后会马上播放
	Controls （控制）	如果出现该属性，则向用户显示控件，比如播放按钮
	loop（循环）	如果出现该属性，则每当音频结束时将重新开始播放
	Preload（加载）	如果出现该属性，则音频在页面加载时进行加载，并预备播放 如果使用 autoplay，则忽略该属性。
	url（地址）	要播放的音频的 URL 地址
autobuffer	Autobuffer（自动缓冲）	在网页显示时，该二进制属性表示是由用户代理（浏览器）自动缓冲的内容，还是由用户使用相关 API 进行内容缓冲

另外，audio 标签可以通过 source 属性添加多个音频文件，具体格式如下：

```
<audio controls="controls">
<source src="123.ogg" type="audio/ogg">
<source src="123.mp3" type="audio/mpeg">
</audio>
```

10.1.3 音频解码器

音频解码器定义了音频数据流编码和解码的算法。其中，编码器主要是对数据流进行编码操作，用于存储和传输。音频播放器主要是对音频文件进行解码，然后进行播放操作。目前，使用较多的音频解码器是 Vorbis 和 ACC。

10.1.4 浏览器对 audio 标签的支持情况

目前，不同的浏览器对 audio 标签的支持也不同。下面的表格中列出应用最为广泛的浏览器对 audio 标签的支持情况。

浏览器 / 音频格式	Firefox 3.5 及更高版本	IE 9.0 及更高版本	Opera 10.5 及更高版本	Chrome 3.0 及更高版本	Safari 3.0 及更高版本
Ogg Vorbis	支持		支持	支持	
MP3		支持		支持	支持
Wav	支持		支持		支持

10.2 video 标签

和音频文件播放方式一样，大多数视频文件在网页上也是通过插件来播放的，例如常见的播放插件为 Flash。由于不是所有的浏览器都拥有同样的插件，所以就需要一种统一的包含视频的标准方法。为此，和 HTML 4 相比，HTML 5 新增了 video 标签。

10.2.1 video 标签概述

video 标签主要是定义播放视频文件或者视频流的标准。支持 3 种视频格式，分别为 Ogg、WebM 和 MPEG 4。

如果需要在 HTML 5 网页中播放视频，输入的基本格式如下：

```
<video src="123.mp4" controls="controls">
</ video >
```

另外，在< video >与</ video >之间插入的内容是供不支持 video 元素的浏览器显示的。

【例 10.2】（实例文件：ch10\10.2.html）

```
<!DOCTYPE html>
<html>
<head>
<title>video</title>
<head>
<body >
<video src="123.mp4" controls="controls">
```

```
    您的浏览器不支持 video 标签！
    </ video >
    </body>
    </html>
```

如果用户的浏览器是 IE 9.0 以前的版本，浏览效果如图 10-3 所示，可见 IE 9.0 以前的浏览器不支持 video 标签。



图 10-3　不支持 video 标签的效果

在 Firefox 8.0 中浏览效果如图 10-4 所示，可以看到加载的视频控制条界面。单击“播放”按钮，即可查看视频的内容。



图 10-4　支持 video 标签的效果

10.2.2　video 标签的属性

video 标签的常见属性和含义如下表所示。

属性	值	描述
autoplay	autoplay	如果出现该属性，则视频在就绪后会马上播放
controls	controls	如果出现该属性，则向用户显示控件，比如播放按钮
	loop	如果出现该属性，则每当视频结束时将重新开始播放
	preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放； 如果使用 autoplay，则忽略该属性
	url	要播放视频的 URL
width	宽度值	设置视频播放器的宽度
height	高度值	设置视频播放器的高度
poster	url	当视频未响应或缓冲不足时，该属性值链接到一个图像。该图像将以一定比例被显示出来

由上表可知，用户可以自定义视频文件显示的大小。例如想让视频以 320×240 像素显示，可以加入 winth 和 height 属性。具体格式如下：

```
<video width="320" height="240" controls src="123.mp4" >
</video>
```

另外，video 标签可以通过 source 属性添加多个视频文件，具体格式如下：

```
<video controls="controls">
<source src="123.ogg" type="video/ogg">
<source src="123.mp4" type="video/mp4">
</ video >
```

10.2.3 视频解码器

视频解码器定义了视频数据流编码和解码的算法。其中，编码器主要是对数据流进行编码操作，用于存储和传输。视频播放器主要是对视频文件进行解码，然后进行播放操作。

目前，在 HTML 5 中，使用比较多的视频解码文件是 Theora、H.264 和 VP8。

10.2.4 浏览器对 video 标签的支持情况

目前，不同的浏览器对 video 标签支持也不同。下面的表格中列出应用最为广泛的浏览器对 video 标签的支持情况。

浏览器 / 视频格式	Firefox 4.0 及更高版本	IE 9.0 及更高版本	Opera 10.6 及更高版本	Chrome 6.0 及更高版本	Safari 3.0 及更高版本
Ogg	支持		支持	支持	
MPEG 4		支持		支持	支持
WebM	支持		支持	支持	

10.3 问题解答

1．在 HTML 5 网页中添加所支持格式的视频，不能在 Firefox 8.0 浏览器中正常播放，为什么？

目前，HTML 5 的 video 标签对视频的支持，不仅仅有视频格式的限制，还有对解码器的限制。规定如下：

- 如果视频是 ogg 格式的文件，则需要带有 Thedora 视频编码和 Vorbis 音频编码的视频软件。
- 如果视频是 MPEG4 格式的文件，则需要带有 H.264 视频编码和 AAC 音频编码的播放软件视频。
- 如果是视频是 WebM 格式的文件，则需要带有 VP8 视频编码和 Vorbis 音频编码的播放软件视频。

2. 在 HTML 5 网页中添加 MP4 格式的视频文件，为什么在不同的浏览器中视频控件显示的外观不同？

在 HTML 5 中规定 controls 属性用来控制视频文件的播放、暂停、停止和调节音量的操作。Controls 是一个布尔属性，一旦添加了此属性，等于告诉浏览器需要显示播放控件并允许用户操作。

因为每一个浏览器负责内置视频控件的外观，所以在不同的浏览器中将显示不同的视频控件外观。

第 11 章　获取地理位置

根据访问者访问网站的方式，有多种获取地理位置的方法，本章主要介绍如何利用 Geolocation API 来获取地理位置。

11.1　用 Geolocation API 获取地理位置

在 HTML 5 网页代码中，通过一些有用的 API，可以查找访问者当前的位置。

11.1.1　地理定位的原理

由于访问者浏览网站的方式不同，可以通过下列方式确定其位置。

- 如果网站浏览者使用计算机上网，可通过获取浏览者的 IP 地址确定其具体位置。
- 如果网站浏览者通过手机上网，可通过获取浏览者的手机信号的接收塔确定其具体位置。
- 如果网站浏览者的设备上具有 GPS 硬件，通过获取 GPS 发出的载波信号获取其具体位置。
- 如果网站浏览者通过无线上网，可以通过无线网络连接获取其具体位置。



提　示

API 是应用程序的编程接口，是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

11.1.2　获取定位信息的方法

在了解了地理定位的原理后，下面介绍获取定位信息的方法，根据访问者访问网站的方式，可以通过下列方法之一确定地理位置。

- 利用 IP 地址定位。
- 利用 GPS 功能定位。
- 利用 Wi-fi 定位。
- 利用 Wi-fi 和 GPRS 联合定位。
- 利用用户自定义定位数据定位。

使用上述的哪种方法将取决于浏览器和设备的功能，然后，浏览器将确定其位置并传输回地理位置，但需要注意的是无法保证返回的位置是设备的实际地理位置。因为，这涉及到隐私

问题，并不是每个人都想共享他的位置。

11.1.3 常用地理定位方法

通过地理定位，可以确定用户的当前位置，并能获取用户地理位置的变化情况。其中，最常用的就是 API 中的 getCurrentpositong 方法。

getCurrentpositong 方法的语法格式如下：

```
void getCurrentPosition(successCallback, errorCallback, options);
```

其中 successCallback 参数是指在成功获取位置时用户想要调用的函数名称，errorCallback 参数是指在位置获取失败时用户想要调用的函数名称，options 参数指出地理定位时的属性设置。


提 示

访问用户位置是耗时的操作，同时出于隐私问题，还要取得用户的同意。

如果地理定位成功，新的 Position 对象将调用 displayOnMap 函数，显示设备的当前位置。

那么 Positon 对象的含义是什么？作为地理定位的 API，Positon 对象包含位置确定时的时间戳（timestamp）和包含位置的坐标（coords），具体语法格式如下：

```
Interface position
 {
readonly attribute Coordinates cords;
readonly attribute DOMTimeStamp timestamp;
}；
```

11.1.4 如何判断浏览器是否支持 HTML 5 获取地理位置信息

在用户试图使用地理定位之前，应该先确保浏览器是否支持 HTML 5 获取地理位置信息。这里介绍判断的方法。具体的代码如下：

```
function init()
if (navigator.geolocation) {
//获取当前地理位置信息
navigator.geolocation.getCurrentPosition(onSuccess, onError, options);
} else {
alert("你的浏览器不支持 html 5 来获取地理位置信息。");
}
```

下面解释该代码中文函数的含义。

1. onSuccess

该函数是获取当前位置信息成功时执行的回调函数。

在 onSuccess 回调函数中，用到了参数 position，代表一个具体的 position 对象，表示当前位置。其具有如下属性。

- latitude：当前地理位置的纬度。
- longitude：当前地理位置的经度。
- altitude：当前位置的海拔高度（不能获取时为 null）。
- accuracy：获取到的纬度和经度的精度（以米为单位）。
- altitudeAccurancy：获取到的海拔高度的经度（以米为单位）。
- heading：设备的前进方向。以前进的方向为正方向顺时针旋转角度来表示（不能获取时为 null）。
- speed：设备的前进速度（以 m/s 为单位，不能获取时为 null）。
- timestamp：获取地理位置信息时的时间。

2. onError

该函数是获取当前位置信息失败时所执行的回调函数。

在 onError 回调函数中，用到了 error 参数，其具有如下属性。

- code：错误代码，有如下值。
  - 用户拒绝了位置服务（属性值为 1）；
  - 获取不到位置信息（属性值为 2）；
  - 获取信息超时错误（属性值为 3）。
- message：字符串，包含了具体的错误信息。

3. options

options 是一些可选熟悉列表。在 options 参数中，可选属性如下。

- enableHighAccuracy：是否要求高精度的地理位置信息。
- timeout：设置超时时间（单位为 ms）。
- maximumAge：对地理位置信息进行缓存的有效时间（单位为 ms）。

11.1.5 指定纬度和经度坐标

对于地理定位成功后，将调用 displayOnMap 函数。此函数如下：

```
function displayOnMap(position)
{
var latitude=positon.coords.latitude;
var longitude=postion.coords.longitude;
}
```

其中第一行函数从 position 对象获取 coordinates 对象，主要由 API 传递给程序调用。第三行和第四行中定义了两个变量，latitude 和 longitude 属性存储在定义的两个变量中。

为了在地图上显示用户的具体位置，可以利用地图网站的 API。下面以使用百度地图为例进行讲解，需要使用 Baidu Maps JavaScript API。在使用此 API 前，需要在 HTML 5 页面中添加一个引用，具体代码如下：

```
<--baidu maps API>
<script type= “text/JavaScript” scr= “http://api.map.baidu.com/api?key=*&v=1.0&services=true” >
</script>
```

其中代“*”号的代码作用是注册到 key。注册 key 的方法为：在“http：//openapi.baidu.com/map/index.html”网页中，注册百度地图 API，然后输入需要内置百度地图页面的 URL 地址，生成 API 密钥，然后将 key 文件复制保存。

虽然已经包含了 Baidu Maps JavaScript，但是页面中还不能显示内置的百度地图，还需要添加 HTML 语言，然后地图从程序转化为对象。还需要加入以下源代码：

```
<script type=“text/JavaScript”scr=“http://api.map.baidu.com/api?key=*&v=1.0&services=true”>
</script>
<div style="width:600px;height:220px;border:1px solid gary;margin-top:15px;" id="container">
</div>
<script type="text/JavaScript">
var map = new BMap.Map("container");
map.centerAndZoom(new BMap.Point(***,***),17);
map.addControl(new BMap.NavigationControl());
map.addControl(new BMap.ScaleControl());
map.addControl(new BMap.OverviewMapControl());
var local = new BMap.LocalSearch(map,
{
enderOptions:{map: map}
}
);
local.search("输入搜索地址");
</script>
```

上述代码分析如下：

（1）其中前 2 行主要是把 baidu map API 程序植入源码中。

（2）第 3 行在页面中设置一个标签，包括宽度和长度，用户可以自己调整；border=1px 是定义外框的宽度为 1 像素，solid 为实线，gray 为边框显示颜色，margin-top 为该标签的上外边距。

（3）第 7 行为在地图中自己位置的坐标。

（4）第 8 行~第 10 行为植入地图缩放控制工具。

（5）第 11 行~第 16 行为地图中自己的位置，只需在 local search 后填入自己的位置名称即可。

【例 11.1】（实例文件：ch11\11.1.html）

如下代码为使用纬度和经度定位坐标的案例。

01 打开记事本文件，在其中输入如下代码：

```
<!DOCTYPE html>
<html>
<head>
<title>纬度和经度坐标</title>
<style>
body {background-color:#fff;}
</style>
</head>
<body>
<p id="geo_loc"><p>
<script>
function getElem(id) {
        return typeof id === 'string' ? document.getElementById(id) : id;
 }

 function show_it(lat, lon) {
        var str = '您当前的位置，纬度：' + lat + '，经度：' + lon;
        getElem('geo_loc').innerHTML = str;
 }
if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position) {
              show_it(position.coords.latitude, position.coords.longitude);
        },
function(err) {
              getElem('geo_loc').innerHTML = err.code + "|" + err.message;
        });
 } else {
        getElem('geo_loc').innerHTML = "您当前使用的浏览器不支持 Geolocation 服务";
 }
</script>
</body>
</html>
```

02 使用 Opera 浏览器打开网页文件，由于使用 HTML 定位功能首先要由用户允许位置共享才可获取地理位置信息，所以弹出如下图所示提示框，选择“总是允许”，单击“确定”按钮，如图 11-1 所示。

03 在弹出的地理位置共享条款对话框中勾选“接受条款和条件并启用地理位置”选项，并单击“接受”按钮，如图 11-2 所示。



图 11-1　程序运行结果


Opera的Geolocation

The Opera browser is attempting to access your location using Google's location service. The service, provided by Google, adds location capability to applications while using the Opera browser.

By clicking agree below, you agree to Google's Terms of Service and to the collection, use, sharing and onward transfer of your location data in accordance with Google's Privacy Policy.

Learn more

接受条款和条件并启用地理位置。

接受　稍后问我

图 11-2　程序运行结果

04 在页面中显示了当前页面打开时所处的地理位置，其位置为使用者的 IP 或 GPS 定位地址，如图 11-3 所示。



图 11-3　程序运行结果



每次使用浏览器打开网页时都会提醒是否允许地理位置共享，为了安全，用户应当妥善使用地址共享功能。

11.2 浏览器对地理定位的支持情况

不用的浏览器版本对地理定位技术的支持情况是不同的，下表是常见浏览器对地理定位的支持情况。

浏览器名称	支持地理定位的浏览器版本
IE	IE 9.0 及更高版本
Firefox	Firefox 3.5 及更高版本
Opera	Opera 10.6 及更高版本
Safari	Safari 5 及更高版本
Chrome	Chrome 5 及更高版本
Android	Android 2.1 及更高版本

11.3 综合实例——在网页中调用 Google 地图

本实例介绍如何在网页中调用 Google 地图，以获取当前设备物理地址的经度与纬度。具体的操作步骤如下：

01 调用 Google Map，其代码如下：

```
<!DOCTYPE html>
<head>
<title>获取当前位置并显示在 google 地图上</title>
<script type="text/JavaScript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/JavaScript">
```

02 获取当前地理位置，其代码如下：

```
navigator.geolocation.getCurrentPosition(function (position) {
var coords = position.coords;
//console.log(position);
```

03 设定地图参数，其代码如下：

```
var latlng = new google.maps.LatLng(coords.latitude, coords.longitude);
var myOptions = {
zoom: 14, //设定放大倍数
center: latlng, //将地图中心点设定为指定的坐标点
mapTypeId: google.maps.MapTypeId.ROADMAP //指定地图类型
```

```
};
```

04 创建地图，并在页面中显示，其代码如下：

```
var map = new google.maps.Map(document.getElementById("map"), myOptions);
```

05 在地图上创建标记，其代码如下：

```
var marker = new google.maps.Marker({
position: latlng, //将前面设定的坐标标注出来
map: map //将该标注设置在刚才创建的 map 中
});
```

06 创建窗体内的提示内容，其代码如下：

```
var infoWindow = new google.maps.InfoWindow({
content: "当前位置：<br/>经度：" + latlng.lat() + "<br/>纬度：" + latlng.lng() //提示窗体内的提示信息
});
```

07 打开提示窗口，其代码如下：

```
infoWindow.open(map, marker);
},
```

08 根据需要再编写其他相关代码，如处理错误的方法和打开地图的大小等。查看此时页面相应的 HTML 源代码如下：

```
<!DOCTYPE html>
<head>
<title>获取当前位置并显示在 google 地图上</title>
<script type="text/JavaScript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/JavaScript">
function init() {
if (navigator.geolocation) {
//获取当前地理位置
navigator.geolocation.getCurrentPosition(function (position) {
var coords = position.coords;
//console.log(position);
//指定一个 Google 地图上的坐标点，同时指定该坐标点的横坐标和纵坐标
var latlng = new google.maps.LatLng(coords.latitude, coords.longitude);
var myOptions = {
zoom: 14, //设定放大倍数
center: latlng, //将地图中心点设定为指定的坐标点
mapTypeId: google.maps.MapTypeId.ROADMAP //指定地图类型
};
//创建地图，并在页面 map 中显示
```

```
var map = new google.maps.Map(document.getElementById("map"), myOptions);
//在地图上创建标记
var marker = new google.maps.Marker({
position: latlng, //将前面设定的坐标标注出来
map: map //将该标注设置在刚才创建的 map 中
});
//标注提示窗口
var infoWindow = new google.maps.InfoWindow({
content: "当前位置：<br/>经度：" + latlng.lat() + "<br/>维度：" + latlng.lng() //提示窗体内的提示信息
});
//打开提示窗口
infoWindow.open(map, marker);
},
function (error) {
//处理错误
switch (error.code) {
case 1:
alert("位置服务被拒绝。");
break;
case 2:
alert("暂时获取不到位置信息。");
break;
case 3:
alert("获取信息超时。");
break;
default:
alert("未知错误。");
break;
}
});
} else {
alert("你的浏览器不支持 HTML 5 来获取地理位置信息。");
}
}
</script>
</head>
<body onload="init()">
<div id="map" style="width: 800px; height: 600px"></div>
</body>
</html>
```

09 保存网页后，即可查看最终效果，如图 11-4 所示。



图 11-4　程序运行结果

11.4　问题解答

1. 使用 HTML 5 Geolocation API 获得的用户地理位置一定精准吗？

不一定精准，因为该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。

2. 地理位置 API 可以在国际空间站上使用吗？可以在月球上或者其他星球上用吗？

地理位置标准是这样阐述的：“地理坐标参考系的属性值来自大地测量系统（World Geodetic System（2d）[WGS84]）。不支持其他参考系。”国际空间站位于地球轨道上，所以宇航员可以使用经纬度和海拔来描述其位置。但是，大地测量系统是以地球为中心的，因此也就不能使用这个系统来描述月球或者其他星球的位置了。

第 12 章 Web 通信新技术

本章主要学习 Web 通信新技术。其中包括跨文档消息传输的实现和 Web Sockets 实时通信技术，通过本章的学习，可以更好地完成跨域数据的通信，以及 Web 即时通信应用的实现，如 Web QQ 等。

12.1 跨文档消息传输

利用跨文档消息传输功能，可以在不同域、端口或网页文档之间进行消息的传递。

12.1.1 跨文档消息传输的基本知识

利用跨文档消息传输可以实现跨域的数据推动，使服务器端不再被动地等待客户端的请求，只要客户端与服务器端建立了一次连接后，服务器端就可以在需要的时候，主动地将数据推送到客户端，直到客户端显示关闭这个连接。

HTML 5 提供了在网页文档之间互相接收与发送消息的功能。使用这个功能，只要获取到网页所在页面对象的实例，不仅同域的 Web 网页之间可以互相通信，甚至可以实现跨域通信。

想要接收从其他的文档那里发过来的消息，就必须对文档对象的 message 时间进行监视，实现代码如下：

```
window.addEventListener(“message”, function(){…}, false)。
```

想要发送消息，可以使用 window 对象的 postMessage 方法来实现，该方法的实现代码如下：

```
otherWindow.postMessage(message, targetOrigin)。
```

说明：postMessage 是 HTML 5 为了解决跨文档通信，特别引入的一个新的 API，目前支持这个 API 的浏览器有：IE（8.0 以上）、Firefox、Opera、Safari 和 Chrome。

postMessage 允许页面中的多个 iframe/window 的通信，postMessage 也可以实现 ajax 直接跨域通信，不通过服务器端代理。

12.1.2 跨文档通信应用测试

下面来介绍一个跨文档通讯的应用案例，其中主要使用 postMessage 的方法来实现该案例。具体操作方法如下：

需要创建两个文档来实现跨文档的访问，名称分别为 12.1.html 和 12.2.html。

01 打开记事本文件，在其中输入以下代码，以创建用于实现信息发送的 12.1.html 文档，具体代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
  <title>跨域文档通信 1</title>
  <meta charset="utf-8"/>
</head>
<script type="text/JavaScript">
    window.onload = function() {
        document.getElementById('title').innerHTML = '页面在' + document.location.host + '域中，且每过 1 秒向 12.2.html 文档发送一个消息！';
//定时向另外一个不确定域的文件发送消息
setInterval(function(){
        var message = '消息发送测试！    ' + (new Date().getTime());
window.parent.frames[0].postMessage(message, '*');
},1000);
    };
</script>
<body>
<div id="title"></div>
</body>
</html>
```

02 保存记事本文件，然后使用浏览器打开该文件，最终的效果如图 12-1 所示。

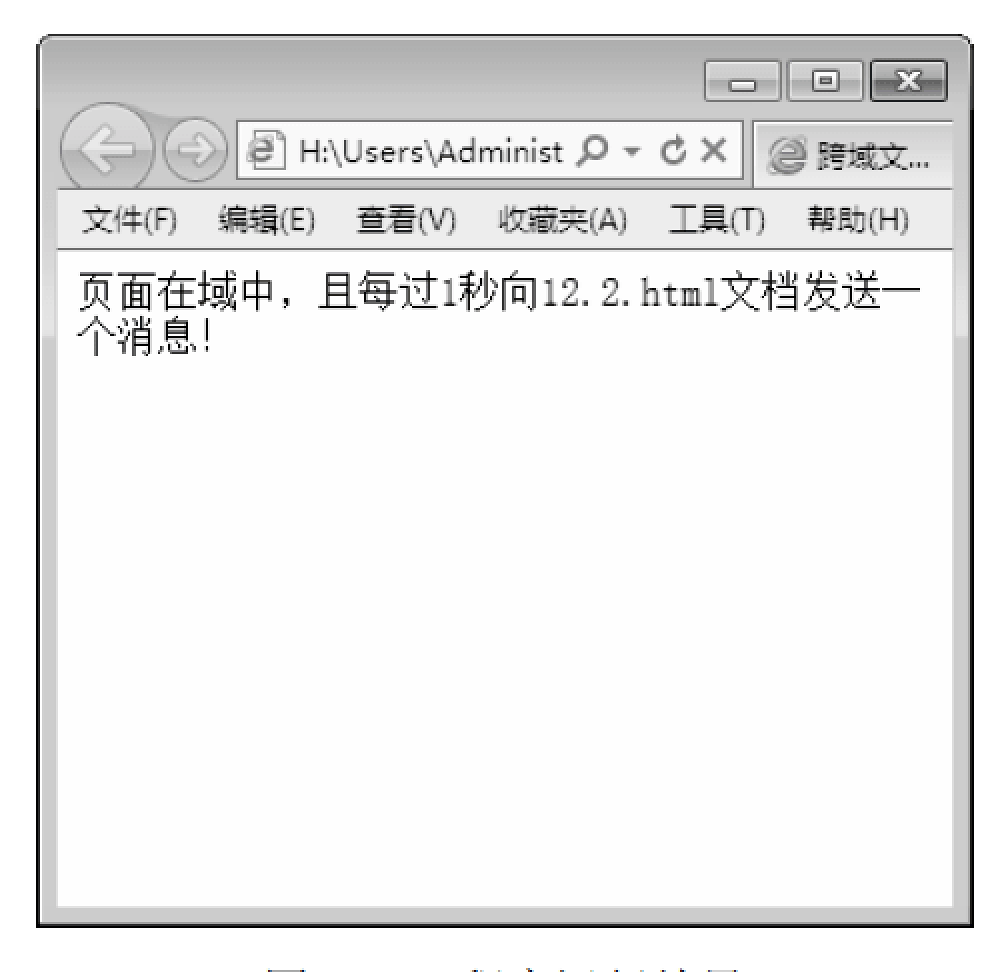


图 12-1 程序运行结果

03 打开记事本文件，在其中输入以下代码，以创建用于实现信息监听的 12.2.html 文档，具体代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
 <title>跨域文档通信 2</title>
 <meta charset="utf-8"/>
</head>
<script type="text/JavaScript">
   window.onload = function() {
      var onmessage = function(e) {
          var data = e.data,p = document.createElement('p');
    p.innerHTML = data;
          document.getElementById('display').appendChild(p);
      };
//监听 postMessage 消息事件
      if (typeof window.addEventListener != 'undefined') {
         window.addEventListener('message', onmessage, false);
      } else if (typeof window.attachEvent != 'undefined') {
         window.attachEvent('onmessage', onmessage);
      }
   };
</script>
<body>
<div id="display"></div>
</body>
</html>
```

运行效果如图 12-2 所示。



图 12-2　程序运行结果

在 12.1.html 文件中的“window.parent.frames[0].postMessage（message, '*'）;”语句中的“*”号表示不对访问的域进行判断。如果要加入特定域的限制，可以将代码改为“window.parent.

frames[0].postMessage（message, 'url'）;” 其中的 url 必须为完整的网站域名格式。而在信息监听接收方的 onmessage 中需要追加一个判断语句 “if（event.origin !== 'url'）return;” 。



提 示

由于在实际通信时，应当实现双向的通信，所以，在编写代码时，每一个文档中都应该具有发送信息和监听接收信息的模块。

12.2 Web Sockets API

HTML 5 中有一个很实用的新特性：WebSockets。使用 WebSockets 可以在没 Ajax 请求的情况下与服务器端对话。

12.2.1 什么是 WebSocket API

WebSocket API 是下一代客户端-服务器的异步通信方法。该通信取代了单个的 TCP 套接字，使用 ws 或 wss 协议，可用于任意的客户端和服务器程序。WebSocket 目前由 W3C 进行标准化。WebSocket 已经受到 Firefox 4、Chrome 4、Opera 10.70 以及 Safari 5 等浏览器的支持。

WebSocket API 设计的最好之处在于服务器和客户端可以在给定时间范围内的任意时刻，相互推送信息。WebSocket 并不限于以 Ajax（或 XHR）方式通信，因为 Ajax 技术需要由客户端发起请求，而 WebSocket 服务器和客户端可以彼此相互推送信息；XHR 受到域的限制，而 WebSocket 允许跨域通信。

Ajax 技术的一点是没有设计要使用的方式。WebSocket 为指定目标创建，用于双向推送消息。

12.2.2 Web Sockets 通信基础

1. 产生 Web Sockets 的背景

随之即时通信系统的普及，基于 Web 的实时通信也变得普及，如新浪微博的评论、私信的通知，腾讯的 Web QQ 等，如图 12-3 所示。



图 12-3 腾讯 Web QQ 页面

在 Web Socket 出现之前，一般通过两种方式来实现 Web 实时应用：轮询机制和流技术，而其中的轮询机制又可分为普通轮询和长轮询（Coment），分别介绍如下：

- 轮询：这是最早的一种实现实时 Web 应用的方案。客户端以一定的时间间隔向服务端发出请求，以频繁请求的方式来保持客户端和服务器端的同步。这种同步方案的缺点是，当客户端以固定频率向服务器发起请求的时候，服务器端的数据可能并没有更新，这样会带来很多无谓的网络传输，所以这是一种非常低效的实时方案。
- 长轮询：是对定时轮询的改进和提高，目地是为了降低无效的网络传输。当服务器端没有数据更新的时候，连接会保持一段时间直到数据或状态改变或者时间过期，通过用这种机制来减少无效的客户端和服务器间的交互。当然，如果服务端的数据变更非常频繁的话，这种机制和定时轮询比较起来没有本质上的性能的提高。
- 流：就是在客户端的页面使用一个隐藏的窗口向服务端发出一个长连接的请求。服务器端接到这个请求后作出回应并不断更新连接状态以保证客户端和服务器端的连接不过期。通过这种机制可以将服务器端的信息源源不断地推向客户端。这种机制在用户体验上有一点问题，需要针对不同的浏览器设计不同的方案来改进用户体验，同时这种机制在并发比较大的情况下，对服务器端的资源是一个极大的考验。

但是上述三种方式实际看来都不是真实的实时通信技术，只是相对的模拟出来实时的效果，这种效果的实现对于编程人员来说无疑增加了复杂性，对于客户端和服务器端的实现都需要复杂的 HTTP 链接设计来模拟双向的实时通信。这种复杂的实现方法制约了应用系统的扩展性。

基于上述弊端，在 HTML 5 中增加了实现 Web 实时应用的技术：Web Socket。Web Socket 通过浏览器提供的 API 真正实现了具备像 C/S 架构下的桌面系统的实时通讯能力。其原理是使用 JavaScript 调用浏览器的 API，API 将发出一个 WebSocket 请求至服务器，经过一次握手和服务器建立了 TCP 通讯，因为它本质上是一个 TCP 连接，所以数据传输的稳定性强和数据传输量小。由于 HTML 5 中 WebSockets 的实用性，使其具备了 Web TCP 的称号。

2. WebSocket 技术的实现方法

WebSocket 技术本质上是一个基于 TCP 的协议技术。其建立通信链接的操作步骤如下：

01 为了建立一个 WebSocket 连接，客户端的浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求有所差异，除了包含一般的头信息外，还有一个附加的信息“Upgrade: WebSocket”，表明这是一个申请协议升级的 HTTP 请求。

02 服务器端解析这些附加的头信息，经过验证后，产生应答信息返回给客户端。

03 客户端接收返回的应答信息，建立与服务器端的 WebSocket 连接，之后双方即可通过该连接通道自由地传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

WebSocket 技术，目前还是属于比较新的技术，其版本更新较快，目前的最新版本基本上可以被 Chrome、FireFox、Opera 和 IE（9.0 以上）等浏览器支持。

在建立实时通信时，客户端发到服务器的内容如下：

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat8.Sec-WebSocket-Version: 13
```

从服务器返回到客户端的内容如下：

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```



其中的“Upgrade:websocket”表示这是一个特殊的 HTTP 请求，请求的目的就是要将客户端和服务器端的通讯协议从 HTTP 协议升级到 WebSocket 协议。其中客户端的 Sec-WebSocket-Key 和服务器端的 Sec-WebSocket-Accept 就是重要的握手认证信息，实现握手后才可以进一步地进行信息的发送和接收。

12.2.3 在服务器端使用 Web Sockets API

在实现 Web Sockets 实时通信时，需要使客户端和服务器端建立链接，需要配置相应的内容，一般构建链接握手时，客户端的内容浏览器都可以代劳完成，主要实现的是服务器端的内容，下面看一下 Web Sockets API 的具体使用方法。

服务器端需要编程人员自己来实现，目前市场上可直接使用的开源方法比较多，主要有以下 5 种。

- Kaazing WebSocket Gateway：是用 Java 实现的 WebSocket Server;
- mod_pywebsocket：是用 Python 实现的 WebSocket Server;
- Netty：是用 Java 实现的网络框架，其中包括了对 WebSocket 的支持;
- node.js：是用 Server 端的 JavaScript 框架，提供了对 WebSocket 的支持;
- WebSocket4Net：是 .net 的服务器端的实现。

除了使用以上开源方法外，程序员编写简单的服务器端也是可以的。其中服务器端需要实现握手、接收和发送三个内容。

下面详细介绍操作方法。

1．握手

在实现握手时需要通过 Sec-WebSocket 信息来验证。使用 Sec-WebSocket-Key 和一个随机值构成一个新的 key 串，然后将新的 key 串进行 SHA1 编码，生成一个由多组两位 16 进制数构成的加密串；最后再把加密串进行 base64 编码生成最终的 key，这个 key 就是 Sec-WebSocket-Accept。

实现 Sec-WebSocket-Key 运算的实例代码如下：

```
///<summary>
/// 生成 Sec-WebSocket-Accept
///</summary>
///<param name="handShakeText">客户端握手信息</param>
///<returns>Sec-WebSocket-Accept</returns>
private static string GetSecKeyAccetp(byte[] handShakeBytes,int bytesLength)
{
    string handShakeText = Encoding.UTF8.GetString(handShakeBytes, 0, bytesLength);
    string key = string.Empty;
    Regex r = new Regex(@"Sec\-WebSocket\-Key:(.*?)\r\n");
    Match m = r.Match(handShakeText);
    if (m.Groups.Count != 0)
    {
key = Regex.Replace(m.Value, @"Sec\-WebSocket\-Key:(.*?)\r\n", "$1").Trim();
    }
    byte[]    encryptionString    =    SHA1.Create().ComputeHash(Encoding.ASCII.GetBytes(key    +
"258EAFA5-E914-47DA-95CA-C5AB0DC85B11"));
    return Convert.ToBase64String(encryptionString);
}
```

2．接收

如果握手成功，将会触发客户端的 onopen 事件，进而解析接收的客户端信息。在进行数据信息解析时，会将数据以字节和比特的方式拆分，并按照以下规则进行解析。

（1）第一字节。

- 第一位：frame-fin，x0 表示该 message 后续还有 frame；x1 表示是 message 的最后一个 frame。
- 第二位~第三位：分别是 frame-rsv1、frame-rsv2 和 frame-rsv3，通常都是 x0。
- 第四位：frame-opcode，x0 表示是延续 frame，x1 表示文本 frame，x2 表示二进制 frame，x3~x7 保留给非控制 frame，x8 表示关闭连接，x9 表示 ping，xA 表示 pong，xB~xF 保留给控制 frame。

（2）第二字节。

- 第一位：Mask，1 表示该 frame 包含掩码；0 表示无掩码。

- 第七位、第七字节和第二字节、第七位和第八字节：第七位取整数值，若在 0~125 之间，则是负载数据长度；若是 126，表示后两个字节取无符号 16 位整数值，是负载长度；127 表示后 8 个字节，取 64 位无符号整数值，是负载长度。

（3）第三字节~第六字节：这里假定负载长度在 0~125 之间，并且 Mask 为 1，则这 4 个字节是掩码。

（4）第七字节和最后一个字节：长度是上面取出的负载长度，包括扩展数据和应用数据两部分，通常没有扩展数据；若 Mask 为 1，则此数据需要解码，解码规则为 1B~4B 掩码循环和数据字节做异或操作。

实现数据解析的代码如下：

```
///<summary>
/// 解析客户端数据包
///</summary>
///<param name="recBytes">服务器接收的数据包</param>
///<param name="recByteLength">有效数据长度</param>
///<returns></returns>
private static string AnalyticData(byte[] recBytes, int recByteLength)
{
    if (recByteLength< 2) { return string.Empty; }
    bool fin = (recBytes[0] & 0x80) == 0x80; // 1bit，1 表示最后一帧
    if (!fin){
return string.Empty;// 超过一帧暂不处理
    }
    bool mask_flag = (recBytes[1] & 0x80) == 0x80; // 是否包含掩码
    if (!mask_flag){
return string.Empty;// 不包含掩码的暂不处理
    }
    int payload_len = recBytes[1] & 0x7F; // 数据长度
    byte[] masks = new byte[4];
    byte[] payload_data;
    if (payload_len == 126){
Array.Copy(recBytes, 4, masks, 0, 4);
payload_len = (UInt16)(recBytes[2]<< 8 | recBytes[3]);
payload_data = new byte[payload_len];
Array.Copy(recBytes, 8, payload_data, 0, payload_len);
    }else if (payload_len == 127){
Array.Copy(recBytes, 10, masks, 0, 4);
byte[] uInt64Bytes = new byte[8];
for (int i = 0; i< 8; i++){
    uInt64Bytes[i] = recBytes[9 - i];
}
```

```
UInt64 len = BitConverter.ToUInt64(uInt64Bytes, 0);
payload  data = new byte[len];
for (UInt64 i = 0; i< len; i++){
      payload_data[i] = recBytes[i + 14];
}
      }else{
Array.Copy(recBytes, 2, masks, 0, 4);
payload_data = new byte[payload_len];
Array.Copy(recBytes, 6, payload_data, 0, payload_len);
      }
      for (var i = 0; i< payload_len; i++){
payload_data[i] = (byte)(payload_data[i] ^ masks[i % 4]);
      }
      return Encoding.UTF8.GetString(payload_data);56.}
```

3. 发送数据

服务器端接收并解析了客户端发来的信息后，要返回相应的信息，服务器发送的数据以 0x81 开头，紧接发送内容的长度，最后是内容的 byte 数组。

实现数据发送的代码如下：

```
///<summary>
/// 打包服务器数据
///</summary>
///<param name="message">数据</param>
///<returns>数据包</returns>
private static byte[] PackData(string message)
{
      byte[] contentBytes = null;
      byte[] temp = Encoding.UTF8.GetBytes(message);
      if (temp.Length< 126){
contentBytes = new byte[temp.Length + 2];
contentBytes[0] = 0x81;
contentBytes[1] = (byte)temp.Length;
Array.Copy(temp, 0, contentBytes, 2, temp.Length);
      }else if (temp.Length< 0xFFFF){
contentBytes = new byte[temp.Length + 4];
contentBytes[0] = 0x81;
contentBytes[1] = 126;
contentBytes[2] = (byte)(temp.Length & 0xFF);
contentBytes[3] = (byte)(temp.Length >>8 & 0xFF);
Array.Copy(temp, 0, contentBytes, 4, temp.Length);
      }else{
```

```
// 暂不处理超长内容
    }
    return contentBytes;
}
```

12.2.4 在客户端使用 Web Sockets API

浏览器提供的 API 就可以直接用来实现客户端的握手操作，在应用时直接使用 JavaScript 来调用即可。

在客户端调用浏览器 API，实现握手操作的 JavaScript 代码如下：

```
var wsServer = 'ws://localhost:8888/Demo';       //服务器地址
var websocket = new WebSocket(wsServer);          //创建 WebSocket 对象
websocket.send("hello");                          //向服务器发送消息
alert(websocket.readyState);                      //查看 WebSocket 当前状态
websocket.onopen = function (evt) {               //已经建立连接
};
websocket.onclose = function (evt) {              //已经关闭连接
};
websocket.onmessage = function (evt) {            //收到服务器消息，使用 evt.data 提取
};
websocket.onerror = function (evt) {              //产生异常
};
```

12.3 综合实例——编写简单的 Web Socket 服务器

在 12.2 节中介绍了 Web Socket API 的原理及基本使用方法，提到在实现通信时关键要配置的是 Web Socket 服务器，下面就来介绍一个简单的 Web Socket 服务器编写方法。

为了实现操作，这里配合编写一个客户端文件，以测试服务器的实现效果。

01 首先编写客户端文件，效果如图 12-4 所示。其文件代码如下：

```
<html>
<head>
<meta charset="UTF-8">
    <title>Web sockets test</title>
    <script src="jquery-min.js" type="text/JavaScript"></script>
    <script type="text/JavaScript">
        var ws;
        function ToggleConnectionClicked() {
            try {
            ws = new WebSocket("ws://192.168.1.101:1818/chat");//连接服务器
ws.onopen = function(event){alert("已经与服务器建立了连接\r\n 当前连接状态："+this.readyState);};
```

```
ws.onmessage = function(event){alert("接收到服务器发送的数据：\r\n"+event.data);};
ws.onclose = function(event){alert("已经与服务器断开连接\r\n 当前连接状态： "+this.readyState);};
ws.onerror = function(event){alert("WebSocket 异常！ ");};
                    } catch (ex) {
              alert(ex.message);
}
          };
          function SendData() {
try{
ws.send("jane");
}catch(ex){
alert(ex.message);
}
          };
function seestate(){
alert(ws.readyState);
}
    </script>
</head>
<body>
 <button id='ToggleConnection' type="button" onclick='ToggleConnectionClicked();'>与服务器建立连接
</button><br /><br />
    <button id='ToggleConnection' type="button" onclick='SendData();'>发送信息：我的名字是
jane</button><br /><br />
    <button id='ToggleConnection' type="button" onclick='seestate();'>查看当前状态</button><br /><br />
</body>
</html>
```



图 12-4　程序运行结果



提 示

其中 ws.onopen、ws.onmessage、ws.onclose 和 ws.onerror 对应了 4 种状态的提示信息。在连接服务器时，需要在代码中指定服务器的链接地址，测试时将 IP 地址改为本机 IP 即可。

02 服务器程序可以使用.net 等实现编辑，编辑后服务器端的主程序代码如下：

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
namespace WebSocket
{
    class Program
    {
        static void Main(string[] args)
        {
            int port = 2828;
            byte[] buffer = new byte[1024];
            IPEndPoint localEP = new IPEndPoint(IPAddress.Any, port);
            Socket listener = new Socket(localEP.Address.AddressFamily,SocketType.Stream,
ProtocolType.Tcp);
            try{
                listener.Bind(localEP);
                listener.Listen(10);
                Console.WriteLine("等待客户端连接....");
                Socket sc = listener.Accept();//接受一个连接
                Console.WriteLine("接受到了客户端："+sc.RemoteEndPoint.ToString()+"连接....");
                //握手
                int length = sc.Receive(buffer);//接受客户端握手信息
                sc.Send(PackHandShakeData(GetSecKeyAccetp(buffer,length)));
                Console.WriteLine("已经发送握手协议了....");
                //接受客户端数据
                Console.WriteLine("等待客户端数据....");
                length = sc.Receive(buffer);//接受客户端信息
                string clientMsg=AnalyticData(buffer, length);
                Console.WriteLine("接受到客户端数据：" + clientMsg);
                //发送数据
                string sendMsg = "您好，" + clientMsg;
                Console.WriteLine("发送数据：“"+sendMsg+"” 至客户端....");
```

```
                sc.Send(PackData(sendMsg));
                Console.WriteLine("演示 Over!");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
……
……
……
        ///<summary>
        /// 打包服务器数据
        ///</summary>
        ///<param name="message">数据</param>
        ///<returns>数据包</returns>
        private static byte[] PackData(string message)
        {
            byte[] contentBytes = null;
            byte[] temp = Encoding.UTF8.GetBytes(message);
            if (temp.Length< 126){
                contentBytes = new byte[temp.Length + 2];
                contentBytes[0] = 0x81;
                contentBytes[1] = (byte)temp.Length;
               Array.Copy(temp, 0, contentBytes, 2, temp.Length);
            }else if (temp.Length< 0xFFFF){
                contentBytes = new byte[temp.Length + 4];
                contentBytes[0] = 0x81;
                contentBytes[1] = 126;
                contentBytes[2] = (byte)(temp.Length & 0xFF);
                contentBytes[3] = (byte)(temp.Length >>8 & 0xFF);
                Array.Copy(temp, 0, contentBytes, 4, temp.Length);
            }else{
                // 暂不处理超长内容
            }
            return contentBytes;
        }
    }
}
```

由于内容较多，这里把中间部分内容省略，编辑后保存到服务器文件目录。

03 测试服务器和客户端的链接通信，首先打开服务器，运行软件包中的“素材

\ch12\12.3\WebSocket-Server\WebSocket\obj\x86\Debug\WebSocket.exe”文件，提示等待客户端链接，效果如图 12-5 所示。

04 运行客户端文件（软件包位置：素材\ch12\12.3\WebSocket-Client\index.html），效果如图 12-6 所示。



图 12-5　程序运行结果



图 12-6　程序运行结果

05 单击“与服务器建立连接”按钮，服务器端显示已经建立链接，客户端提示连接建立，且状态为 1，效果如图 12-7 所示。

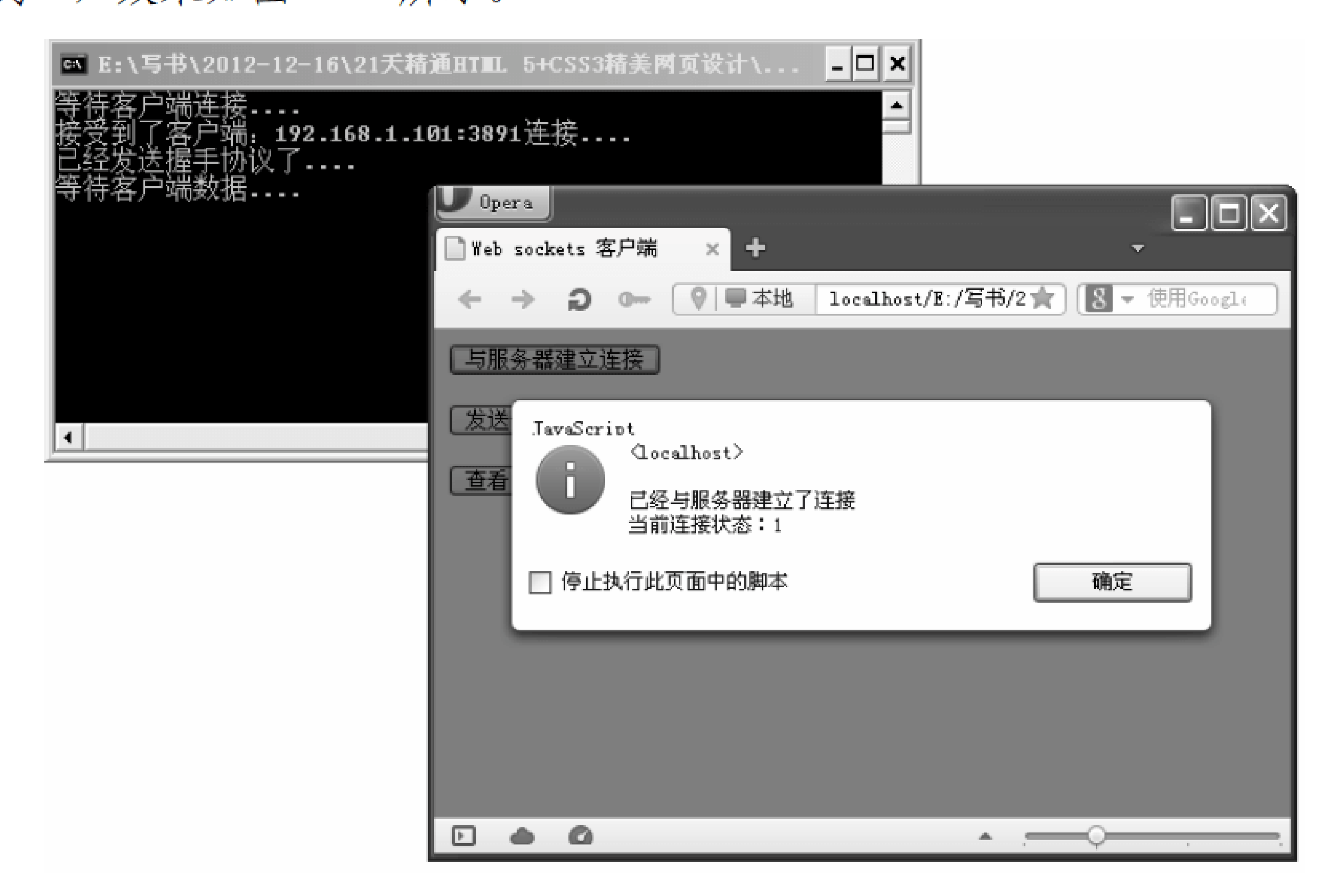


图 12-7　程序运行结果

06 单击“发送消息”按钮，自服务器端返回信息，提示“您好，jane”。



图 12-8　程序运行结果

12.4　问题解答

1. WebSockets 能替代什么？

WebSockets 可以替代 Long Polling（PHP 服务端推送技术）。客户端发送一个请求到服务器，服务器端并不会响应还没准备好的数据，它会保持连接的打开状态直到最新的数据准备就绪再发送，之后客户端收到数据，然后发送另一个请求。好处在于减少任一连接的延迟，当一个连接已经打开时就不需要创建另一个新的连接了。但是 Long-Polling 并不是什么花俏技术，仍有可能发生请求暂停，因此会需要建立新的连接。

2. WebSocket 的优势在哪里？

它可以实现真正的实时数据通信。众所周知，B/S 模式下应用的是 HTTP 协议，是无状态的，所以不能保持持续的连接。数据交换是通过客户端提交请求到服务器端，然后服务器端返回一个应答到客户端来实现的。而 WebSocket 是通过 HTTP 协议的初始握手阶段然后升级到 Web Socket 协议以支持实时数据通信。

WebSocket 可以支持服务器主动向客户端推送数据。一旦服务器和客户端通过 WebSocket 建立起连接，服务器便可以主动的向客户端推送数据，而不像普通的 Web 传输方式需要先由客户端发送请求才能返回数据，从而增强了服务器的能力。

WebSocket 协议设计了更为轻量级的 Header，除了首次建立连接时需要发送头部和普通 Web 连接类似的数据之外，建立 WebSocket 连接后，相互沟通的 Header 就会异常的简洁，大大减少了冗余的数据传输。

WebSocket 提供了更为强大的通信能力和更为简洁的数据传输平台，能更为方便的完成 Web 开发中的双向通信功能。

第 13 章　本地存储技术

Web Storage 是 HTML 5 引入的一个非常重要的功能，可以在客户端本地存储数据，类似 HTML 4 的 Cookie，但可实现功能要比 Cookie 强大得多，Cookie 大小被限制为 4KB，Web Storage 官方建议为每个网站 5MB。

13.1　认识 Web 存储

在 HTML 5 标准之前，Web 存储信息需要 Cookie 来完成，但是 Cookie 不适合大量数据的存储情况，因为数据是靠对服务器的请求来传递的，这使得 Cookie 速度很慢而且效率也不高。为此，在 HTML 5 中，Web 存储 API 为用户如何在计算机或设备上存储用户信息作了数据标准的定义。

13.3.1　本地存储和 Cookie 的区别

本地存储和 Cookie 扮演着类似的角色，但是它们有根本的区别。

- 本地存储仅将数据存储在用户的硬盘上，等待用户读取，而 Cookie 将数据存储在服务器上。
- 本地存储仅供客户端使用，如果需要服务器根据存储数值作出反映，就应该使用 Cookie。
- 读取本地存储不会影响到网络带宽，但是使用 Cookie 将会发送数据到服务器，这样会影响到网络带宽，无形中增加了成本。
- 从存储容量上看，本地存储可存储多达 5MB 的数据，而 Cookie 最多只能存储 4KB 的数据信息。

13.3.2　Web 存储方法

在 HTML 5 标准中，提供了以下两种在客户端存储数据的新方法。

- sessionStorage：是基于 session 的数据存储，在关闭或者离开网站后，数据将会被删除，也被称为会话存储。
- localStorage：没有时间限制的数据存储，也被称为本地存储。

与会话存储不用，本地存储将在用户计算机上永久保持数据信息。关闭浏览器窗口后，如果再次打开该站点，可以检索所有存储在本地的数据。

在 HTML 5 中，数据不是由每个服务器请求传递的，而是只有在请求时使用数据，这样的话，存储大量数据时不会影响网站性能。对于不同的网站，数据存储于不同的区域，并且网站只能访问其自身的数据。



HTML 5 使用 JavaScript 来存储和访问数据，为此，建议用户多了解 JavaScript 的基本知识。

13.2 HTML 5 Web Storage API

使用 HTML 5 Web Storage API 技术，可以很好地实现本地存储。

13.2.1 测试浏览器的支持情况

Web Storage 在各大主流浏览器中都被支持，但是为了兼容老版浏览器，还是要检查一下是否可以使用这项技术，主要有两种方法。

1. 通过检查 Storage 对象来判断

第一种方式：通过检查 Storage 对象是否存在来检查浏览器是否支持 Web Storage，代码如下：

```
if(typeof(Storage)!=="undefined"){
// Yes! localStorage and sessionStorage support!
// Some code.....
} else {
// Sorry! No web storage support..
}
```

2. 分别检查各自的对象

第二种方式就是分别检查各自的对象，例如：检查 localStorage 是否支持，代码如下：

```
if (typeof(localStorage) == 'undefined' ) {
alert('Your browser does not support html 5 localStorage. Try upgrading.');
} else {
// Yes! localStorage and sessionStorage support!
// Some code.....
}
或者：
if('localStorage' in window && window['localStorage'] !== null){
// Yes! localStorage and sessionStorage support!
// Some code.....
} else {
alert('Your browser does not support html 5 localStorage. Try upgrading.');
}
```

```
或者
if (!!localStorage) {
// Yes! localStorage and sessionStorage support!
// Some code.....
} else {
alert('Your browser does not support html 5 localStorage. Try upgrading.');
}
```

13.2.2 sessionStorage 方法

sessionStorage 方法针对 session 进行数据存储。如果用户关闭浏览器窗口后，数据会被自动删除。

创建 sessionStorage 方法的基本语法格式如下：

```
<script type="text/JavaScript">
sessionStorage.abc="    ";
</script>
```

1. 创建对象

【例 13.1】（实例文件：ch13\13.1.html）

```
<!DOCTYPE html>
<html>
<body>
<script type="text/JavaScript">
sessionStorage.name="我们的公司是:英达科技文化公司";
document.write(sessionStorage.name);
</script>
</body>
</html>
```

在 IE 9.0 中浏览效果如图 13-1 所示，即可看到 sessionStorage 方法创建的对象内容显示在网页中。



图 13-1 用 sessionStorage 方法创建对象的效果

2. 制作网站访问记录计数器

下面继续使用 sessionStorage 方法做实例，主要制作记录用户访问网站次数的计数器。

【例 13.2】（实例文件：ch13\13.2.html）

```
<!DOCTYPE html>
<html>
<body>
<script type="text/JavaScript">
if (sessionStorage. count)
{
sessionStorage.count=Number(sessionStorage.count) +1;
}
else
{
sessionStorage. count=1;
}
document.write("您访问该网站的次数为：" + sessionStorage.count);
</script>
</body>
</html>
```

在 IE 9.0 中浏览效果如图 13-2 所示。如果用户刷新一次页面，计数器的数值将加 1。

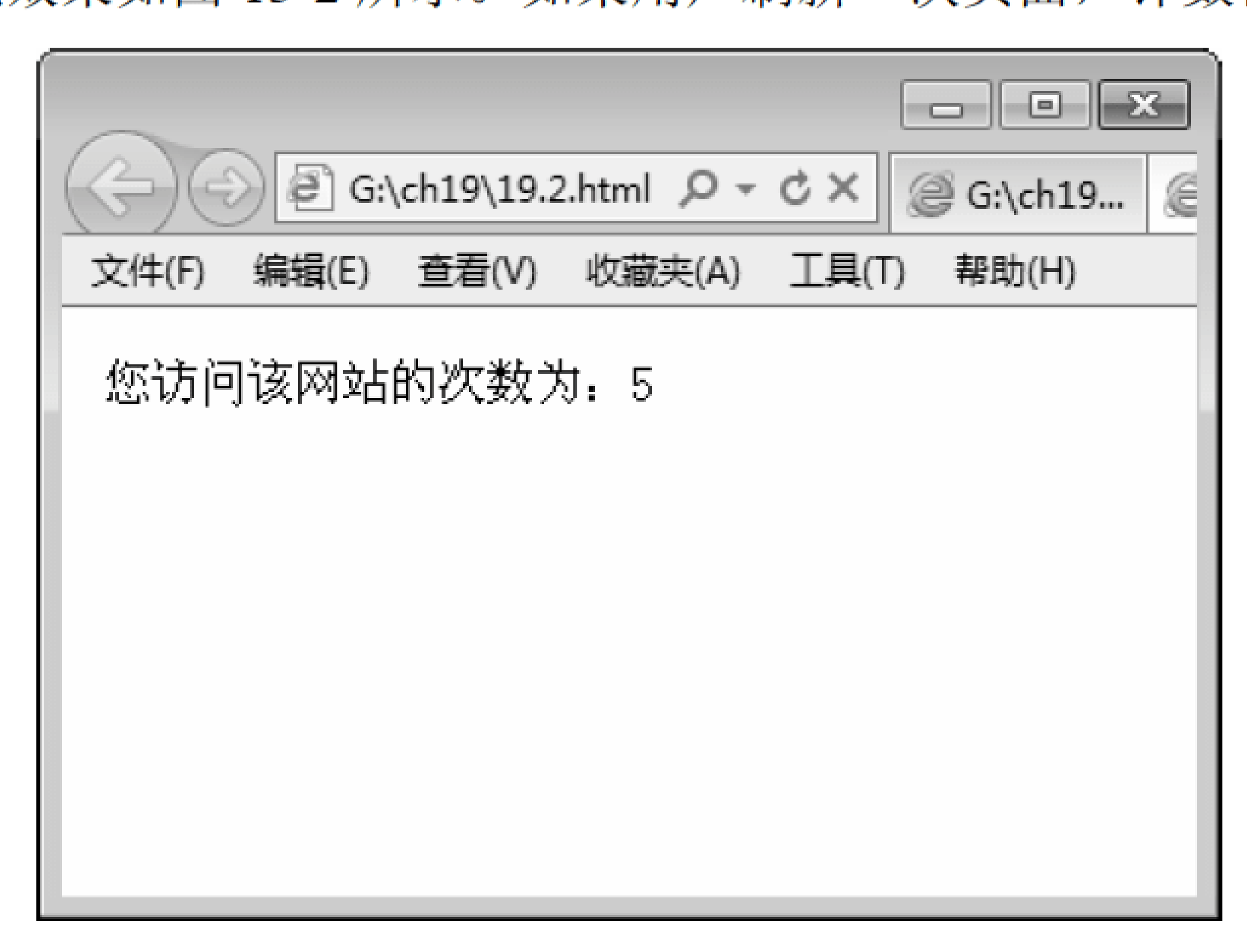


图 13-2　用 sessionStorage 方法创建计数器效果



如果用户关闭浏览器窗口，再次打开该网页时，计数器将重置为 1。

13.2.3　localStorage 方法

与 seessionStorage 方法不同，localStorage 方法存储的数据没有时间限制。也就是说网页浏览者关闭网页后，再次打开此网页时，数据依然可用。

创建一个 localStorage 方法的基本语法格式如下：

```
<script type="text/JavaScript">
localStorage.abc="    ";
</script>
```

1. 创建对象

【例 13.3】（实例文件：ch13\13.3.html）

```
<!DOCTYPE html>
<html>
<body>
<script type="text/JavaScript">
localStorage.name="学习 html 5 最新的技术：Web 存储";
document.write(localStorage.name);
</script>
</body>
</html>
```

在 IE 9.0 中浏览效果如图 13-3 所示，即可看到 localStorage 方法创建的对象内容显示在网页中。



图 13-3　用 localStorage 方法创建对象的效果

2. 制作网站访问记录计数器

下面仍然使用 localStorage 方法来制作记录用户访问网站次数的计数器。用户可以清楚地看到 localStorage 方法和 sessionStorage 方法的区别。

【例 13.4】（实例文件：ch13\13.4.html）

```
<!DOCTYPE html>
<html>
<body>
<script type="text/JavaScript">
if (localStorage.count)
{
localStorage.count=Number(localStorage.count) +1;
}
else
{
localStorage.count=1;
 }
document.write("您访问该网站的次数为：" + localStorage.count");
</script>
</body>
</html>
```

在 IE 9.0 中浏览效果如图 13-4 所示。如果用户刷新一次页面，计数器的数值将进行加 1；如果用户关闭浏览器窗口，再次打开该网页，计数器会继续上一次计数，而不会重置为 1。



图 13-4　用 localStorage 方法创建计数器效果

13.2.4　Web Storage API 的其他操作

Web Storage API 的 localStorage 和 sessionStorage 对象除了以上基本应用外，还有以下两个用法。

1. 清空 localStorage 数据

localStorage 的 clear()函数用于清空同源的本地存储数据，比如 localStorage.clear()，它将

删除所有本地存储的 localStorage 数据。

而 Web Storage 的另外一部分 Session Storage 中的 clear()函数只清空当前会话存储的数据。

2. 遍历 localStorage 数据

遍历 localStorage 数据可以查看 localStrage 对象保存的全部数据信息。在遍历过程中，需要访问 localStorage 对象的另外两个属性 length 与 key。length 表示 localStorage 对象中保存数据的总量，key 表示保存数据时的键名项，该属性常与索引号（index）配合使用，表示键名对应的数据记录，其中，索引号（index）以 0 值开始，如果取第 3 条键名对应的数据，index 值应该为 2。

取出数据并显示数据内容的代码命令如下：

```
functino showInfo(){
var array=new Array();
for(var i=0;i
//调用 key 方法获取 localStorage 中数据对应的键名
//如这里键名是从 test1 开始递增到 testN 的，那么 localStorage.key(0)对应 test1
var getKey=localStorage.key(i);
//通过键名获取值，这里的值包括内容和日期
var getVal=localStorage.getItem(getKey);
//array[0]就是内容，array[1]是日期
array=getVal.split(",");
}
}
```

获取并保存数据的代码命令如下：

```
var storage = window.localStorage; f
or (var i=0, len = storage.length; i <    len; i++){
var key = storage.key(i);
var value = storage.getItem(key);
console.log(key + "=" + value); }
```



由于 localStorage 不仅存储了这里所添加的信息，可能还存储其他信息，但是那些信息的键名也是以递增数字形式表示的，这样如果这里也用纯数字就可能覆盖另外一部分的信息，所以建议键名都用独特的字符区分开，这里在每个 ID 前加上 test 以示区别。

3. 使用 JSON 对象存取数据

在 HTML 5 中可以使用 JSON 对象存取一组相关的对象。使用 JSON 对象可以收集一组用户输入信息，创建 Object 来囊括这些信息，用 JSON 字符串来表示这个 Object，把 JSON 字符

串存放在 localStorage 中。当用户检索指定名称时，会自动用该名称去 localStorage 取得对应的 JSON 字符串，将字符串解析到 Object 对象，然后依次提取对应的信息，并构造 HTML 文本输入显示。

【例 13.5】（实例文件：ch13\13.5.html）

下面列举一个简单的案例，来介绍如何使用 JSON 对象存取数据，具体操作方法如下：

01 新建一个记事本文件，具体代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>使用 JSON 对象存取数据</title>
<script type="text/JavaScript" src="objectStorage.js"></script>
</head>
<body>
<h3>使用 JSON 对象存取数据</h3>
<h4>填写待存取信息到表格中</h4>
<table>
<tr><td>用户名:</td><td><input type="text" id="name"></td></tr>
<tr><td>E-mail:</td><td><input type="text" id="email"></td></tr>
<tr><td>联系电话:</td><td><input type="text" id="phone"></td></tr>
<tr><td></td><td><input type="button" value="保存" onclick="saveStorage();"></td></tr>
</table>
<hr>
<h4>检索已经存入 localStorage 的 json 对象，并且展示原始信息</h4>
<p>
<input type="text" id="find">
<input type="button" value="检索" onclick="findStorage('msg');">
</p>
<!-- 下面这块用于显示被检索到的信息文本 -->
<p id ="msg"></p>
</body>
</html>
```

02 使用 IE 9.0 浏览保存的 html 文件，页面显示效果如图 13-5 所示。



图 13-5 创建存取对象的表格

03 案例中用到了 JavaScript 脚本，其中包含两个函数，一个存数据，一个取数据，具体的 JavaScript 脚本代码如下:

```
function saveStorage(){                //创建一个 js 对象，用于存放当前从表单获得的数据
var data = new Object;                 //将对象的属性值名依次和用户输入的属性值关联起来
data.user=document.getElementById("user").value;
data.mail=document.getElementById("mail").value;
data.tel=document.getElementById("tel").value;
//创建 json 对象，让其与在 html 文件中创建的对象的字符串数据形式对应
var str = JSON.stringify(data);
//将 json 对象存放到 localStorage 上，key 为用户输入的 NAME，value 为这个 json 字符串
localStorage.setItem(data.user,str);
console.log("数据已经保存！被保存的用户名为: "+data.user);
}
//从 localStorage 中检索用户输入的名称对应的 json 字符串，然后把 json 字符串解析为一组信息，并且打印到指定位置
function findStorage(id){                 //获得用户的输入，是用户希望检索的名字
var requiredPersonName = document.getElementById("find").value;
//以这个检索的名字来查找 localStorage，得到 json 字符串
var str=localStorage.getItem(requiredPersonName);
//解析这个 json 字符串得到 Object 对象
var data= JSON.parse(str);
//从 Object 对象中分离出相关属性值，然后构造要输出的 html 内容
var result="用户名:"+data.user+'<br>';
result+="E-mail:"+data.mail+'<br>';
result+="联系电话:"+data.tel+'<br>';              //取得页面上要输出的容器
var target = document.getElementById(id);          //用刚才创建的 html 内容来填充这个容器
target.innerHTML = result;
```

```
}
```

04 将 js 文件和 html 文件放在同一目录下，再次打开网页，在表单中依次输入相关内容，单击“保存”按钮，如图 13-6 所示。



图 13-6　输入表格内容

05 在“检索”文本框中输入已经保存的信息的用户名，单击“检索”按钮，则在页面下方自动显示保存的用户信息，如图 13-7 所示。

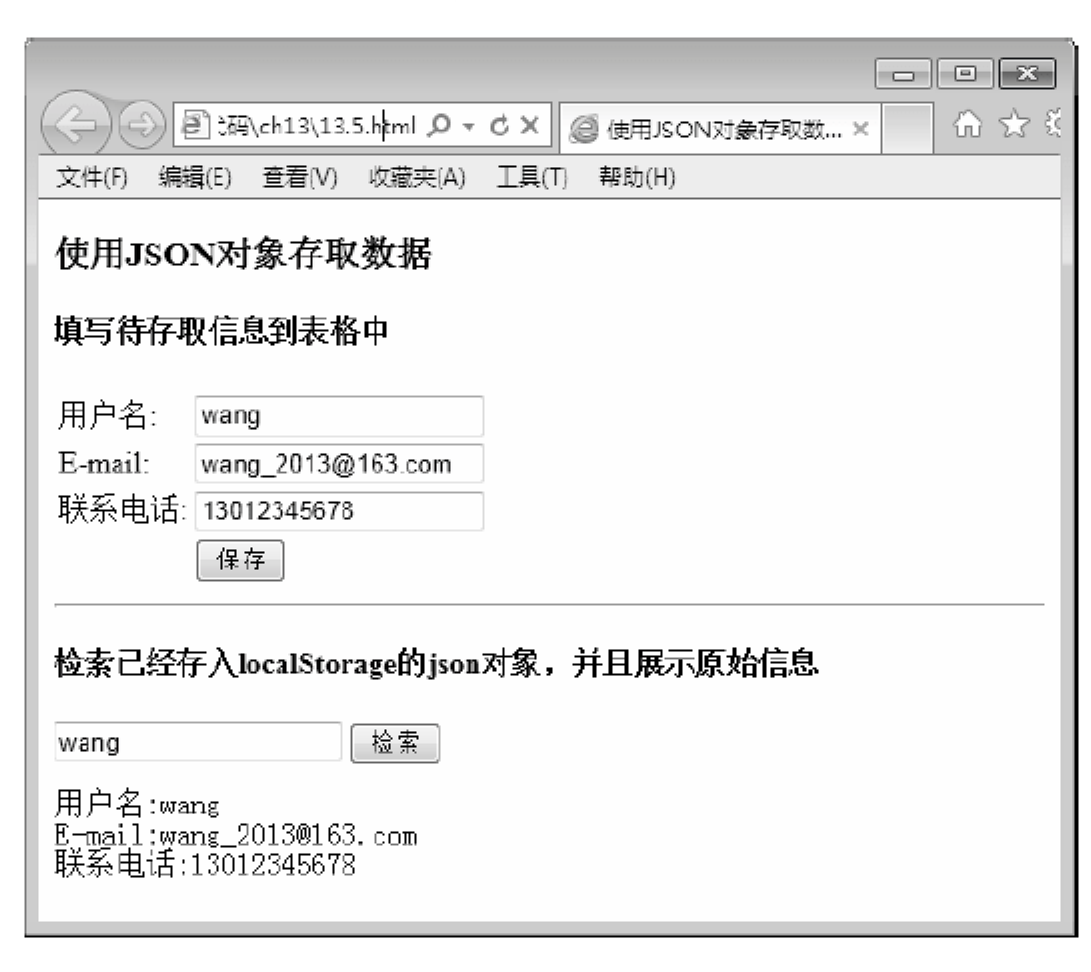


图 13-7　检索数据信息

13.3　在本地建立数据库

上述简单介绍了如何利用 localStorage 实现本地存储；实际上，除了 sessionStorage 和 localStorage 外，HTML 5 还支持通过本地数据库进行本地数据存储，HTML 5 采用的是 SQLLite 这种文件类型数据库，该数据库多集中在嵌入式设备上。

13.3.1 本地数据库概述

可以使用 OpenDatabase 方法打开已经存在的数据库，如果数据库不存在，则使用此方法将会创建新数据库。打开或创建数据库的代码命令如下：

```
var db = openDatabase('mydb', '1.1', ' A list of to do items.', 200000);
```

上述代码的括号中设置了 5 个参数，其意义分别为：数据库名称、版本号、文字说明、数据库的大小和创建回滚。



如果数据库已经创建了，第五个参数将会调用此回滚操作。如果省略此参数，则仍将创建正确的数据库。

以上代码的意义：创建了一个数据库对象 db，名称是 mydb，版本编号为 1.1。db 还带有描述信息和大概的大小值。用户代理（user agent）可使用这个描述与用户进行交流，说明数据库是用来做什么的。利用代码中提供的大小值，用户代理可以为内容留出足够的存储。如果需要，这个大小是可以改变的，所以没有必要预先假设允许用户使用多少空间。

为了检测之前创建的连接是否成功，可以检查那个数据库对象是否为 null：

```
if(!db)
    alert("Failed to connect to database.");
```

绝不可以假设该连接已经成功建立，即使过去对于某个用户是成功的。为什么连接会失败，存在多个原因。也许用户代理出于安全原因拒绝你的访问，也许设备存储有限。面对活跃而快速进化的潜在用户代理，对用户的机器、软件及其能力作出假设是非常不明智的行为。

13.3.2 用 executeSql 来执行查询

通过 executeSql 方法执行查询，代码如下：

```
tx.executeSql(sqlQuery,[value1,value2..],dataHandler,errorHandler)
```

executeSql 方法有 4 个参数，作用分别如下。

- sqlQuery：需要具体执行的 sql 语句，可以是 create、select、update、delete;
- [value1,value2..]：sql 语句中所有使用到的参数数组，在 executeSql 方法中，将 sql 语句中所要使用的参数先用“?”代替，然后依次将这些参数组成数组放在第二个参数中；
- dataHandler：执行成功是调用的回调函数，通过该函数可以获得查询结果集;
- errorHandler：执行失败时调用的回调函数。

13.3.3 使用 transaction 方法处理事件

使用第一步创建的数据库访问对象（如 db）执行 transaction 方法，用来执行事务处理，代码如下：

```
db.transaction(function(tx)){
//执行访问数据库的语句
});
```

transaction 方法使用一个回调函数作为参数，在这个函数中，执行访问数据库的具体操作。

13.4 浏览器对 Web 存储的支持情况

不同的浏览器版本对 Web 存储技术的支持情况是不同的，下表是常见浏览器对 Web 存储的支持情况。

浏览器名称	支持 Web 存储技术的浏览器版本
IE	IE 8 及更高版本
Firefox	Firefox 3.6 及更高版本
Opera	Opera 10.0 及更高版本
Safari	Safari 4 及更高版本
Chrome	Chrome 5 及更高版本
Android	Android 2.1 及更高版本

13.5 综合实例——制作简单 Web 留言本

使用 Web Storage 功能可以用来制作 Web 留言本，具体制作方法如下：

01 构建页面框架，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>本地存储技术之 Web 留言本</title>
</head>
<body onload="init()">
</body>
</html>
```

02 添加页面文件，主要由表单构成，包括单行文本表单和多行文本表单，代码如下：

```
<h1>Web 留言本</h1>
<table>
    <tr>
        <td>用户名</td>
        <td><input type="text" name="name" id="name" /></td>
    </tr>
```

```
        <tr>
            <td>留言</td>
            <td><textarea name="memo" id="memo" cols ="50" rows = "5"></textarea></td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type="submit" value="提交" onclick="saveData()" />
            </td>
        </tr>
    </table>
    <ht>
    <table id="datatable" border="1"></table>
    <p id="msg"></p>
```

03 为了执行本地数据库的保存及调用功能，需要插入数据库的脚本代码，具体内容如下：

```
<script>
var datatable = null;
var db = openDatabase("MyData","1.0","My Database",2*1024*1024);
function init()
{
    datatable = document.getElementById("datatable");
    showAllData();
}
function removeAllData(){
    for(var i = datatable.childNodes.length-1;i>=0;i--){
        datatable.removeChild(datatable.childNodes[i]);
    }
    var tr = document.createElement('tr');
    var th1 = document.createElement('th');
    var th2 = document.createElement('th');
    var th3 = document.createElement('th');
    th1.innerHTML = "用户名";
    th2.innerHTML = "留言";
    th3.innerHTML = "时间";
    tr.appendChild(th1);
    tr.appendChild(th2);
    tr.appendChild(th3);
    datatable.appendChild(tr);
}
function showAllData()
```

```
{
    db.transaction(function(tx){
        tx.executeSql('create table if not exists MsgData(name TEXT,message TEXT,time INTEGER)',[]);
        tx.executeSql('select * from MsgData',[],function(tx,rs){
            removeAllData();
            for(var i=0;i<rs.rows.length;i++){
                showData(rs.rows.item(i));
            }
        });
    });
}
function showData(row){
    var tr=document.createElement('tr');
    var td1 = document.createElement('td');
    td1.innerHTML = row.name;
    var td2 = document.createElement('td');
    td2.innerHTML = row.message;
    var td3 = document.createElement('td');
    var t = new Date();
    t.setTime(row.time);
    ttd3.innerHTML = t.toLocaleDateString() + " " + t.toLocaleTimeString();
    tr.appendChild(td1);
    tr.appendChild(td2);
    tr.appendChild(td3);
    datatable.appendChild(tr);
}
function addData(name,message,time) {
    db.transaction(function(tx){
        tx.executeSql('insert into MsgData values(?,?,?)',[name,message,time],functionx,rs){
            alert("提交成功。");
        },function(tx,error){
            alert(error.source+"::"+error.message);
        });
    });
} // End of addData
function saveData() {
    var name = document.getElementById('name').value;
    var memo = document.getElementById('memo').value;
    var time = new Date().getTime();
    addData(name,memo,time);
    showAllData();
} // End of saveData
```

```
</script>
</head>
<body onload="init()">
    <h1>Web 留言本</h1>
    <table>
        <tr>
            <td>用户名</td>
            <td><input type="text" name="name" id="name" /></td>
        </tr>
        <tr>
            <td>留言</td>
            <td><textarea name="memo" id="memo" cols ="50" rows = "5"></textarea></td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type="submit" value="提交" onclick="saveData()" />
            </td>
        </tr>
    </table>
    <ht>
    <table id="datatable" border="1"></table>
    <p id="msg"></p>
</body>
</html>
```

04 文件保存后，使用 IE 9.0 浏览页面，效果如图 13-8 所示。

图 13-8 Web 留言本

13.6 问题解答

1．不同的浏览器可以读取同一个 Web 中存储的数据吗？

在 Web 存储时，不同的浏览器将存储在不同的 Web 存储库中。例如用户使用的是 IE 浏览器，那么 Web 存储工作时，将所有数据存储在 IE 的 Web 存储库中，如果用户使用火狐浏览器访问该站点，将不能读取 IE 浏览器存储的数据，可见每个浏览器的存储是分开并独立工作的。

2. 离线存储站点时是否需要浏览者同意？

和地理定位类似，在网站使用 manifest 文件时，浏览器会提供一个权限提示，提示用户是否将离线设为可用，但是不是每个浏览器都支持这样的操作。

第 14 章　线程处理技术

利用 Web worker 技术，可以实现网页脚本程序的多线程后台执行，并且不会影响其他脚本的执行，为大型网站的顺畅运行提供了更好的实现方法。本章主要讲述线程处理技术，包括 Web Workers 概述和线程的嵌套运用等。

14.1　Web Workers

在 HTML 5 中为了提供更好的后台程序，设计了 Web Worker 技术。Web Worker 的产生主要是考虑到在 HTML 4 中执行的 JavaScript Web 程序都是以单线程的方式执行的，一旦前面的脚本花费时间过长，后面的程序就会因长期得不到响应而使用户页面操作出现异常。

14.1.1　Web Workers 概述

Web Worker 实现的是线程技术，可以使运行在后台的 JavaScript 独立于其他脚本，不会影响页面的性能。

Web Worker 创建后台线程的方法非常简单，只需要将在后台线程中执行的脚本文件，以 URL 地址的方式创建在 worker 类的构造器中就可以了，其代码格式如下：

```
var worker=new worker(“worker.js”);
```

目前大部分主流的浏览器都支持 Web Worker 技术。创建 Web Worker 之前，用户可以检测浏览器是否支持它，可以使用以下方法检测浏览器对 Web Worker 的支持情况：

```
if(typeof(Worker)!=="undefined")
  {
  // Yes! Web Worker support!
  // Some code.....
  }
else
  {
  // Sorry! No Web Worker support..
  }
```

如果浏览器不支持该技术的话，将会出现如下提示信息，如图 14-1 所示。

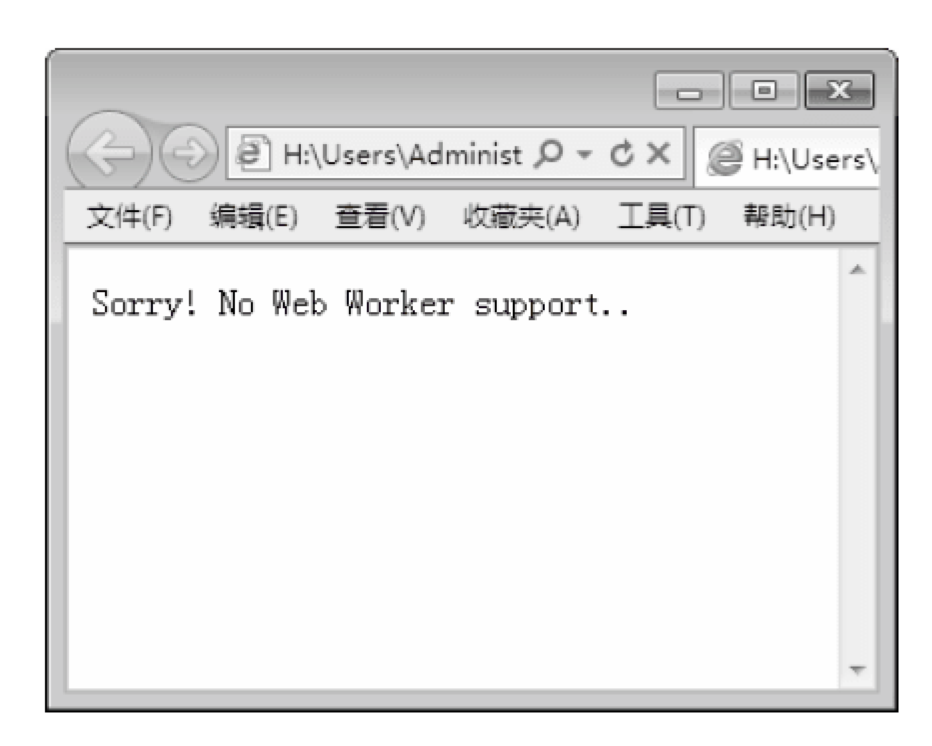


图 12-1 程序运行结果

14.1.2 线程中常用的变量、函数与类

在进行 Web Worker 线程创建时会涉及到一些变量、函数与类内容，其中在线程中执行的 JavaScript 脚本文件中可以用到的变量、函数与类介绍如下。

- Self：Self 关键词用来表示本线程范围内的作用域。
- Imports：导入的脚本文件必须与使用该线程文件的页面在同一个域中，并在同一个端口中。
- ImportScripts（urls）：导入其他 JavaScript 脚本文件。参数为该脚本文件的 URL 地址，可以导入多个脚本文件。
- Onmessage：获取接收消息的事件句柄。
- Navigator 对象：与 window.navigator 对象类似，具有 appName、platform、userAgent、appVersion 这些属性。
- setTimeout()/setInterval()：可以在线程中实现定时处理。
- XMLHttpRequest：可以在线程中处理 Ajax 请求。
- Web Workers：可以在线程中嵌套线程。
- SessionStorage/localStorage：可以在线程中使用 Web Storage
- Close：可以结束本线程。
- Eval()、isNaN()、escape()：可以使用所有 JavaScript 核心函数。
- Object：可以创建和使用本地对象。
- WebSockets：可以使用 WebSockets API 来向服务器发送和接收信息。
- postMessage（message）：向创建线程的源窗口发送消息。

14.1.3 与线程进行数据的交互

在后台执行的线程是不可以访问页面和窗口对象的，但这并不妨碍前台和后台线程进行数据的交互，下面就来介绍一个前台和后台线程交互的案例。

本案例中，后台执行的 JavaScript 脚本线程从 0~200 的所有整数中随机挑选一些整数，然后在这些选出的整数中选择可以被 5 整除的整数，最后将这些选出的整数交给前台显示，以实现前台与后台线程的数据交互。

01 完成前台的网页代码，其代码内容如下（详见代码包中的“素材\ch14\14.1.html”）：

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<title>前台与后台线程的数据交互</title>
<script type="text/JavaScript">
var intArray=new Array(200);//随机数组
var intStr="";        //将随机数组用字符串进行连接
//生成 200 个随机数
for(var i=0;i<200;i++)
{
    intArray[i]=parseInt(Math.random()*200);
    if(i!=0)
        intStr+=";";          //用分号作随机数组的分隔符
    intStr+=intArray[i];
}
//向后台线程提交随机数组
var worker = new Worker("14.1.js");
worker.postMessage(intStr);
// 从线程中取得计算结果
worker.onmessage = function(event) {
    if(event.data!="")
    {
        var h;                    //行号
        var l;                    //列号
        var tr;
        var td;
        var intArray=event.data.split(";");
        var table=document.getElementById("table");
        for(var i=0;i<intArray.length;i++)
        {
            h=parseInt(i/15,0);
            l=i%15;
            //该行不存在
            if(l==0)
            {
                //添加新行的判断
                tr=document.createElement("tr");
                tr.id="tr"+h;
                table.appendChild(tr);
            }
```

```
                    //该行已存在
                    else
                    {
                        //获取该行
                        tr=document.getElementById("tr"+h);
                    }
                    //添加列
                    td=document.createElement("td");
                    tr.appendChild(td);
                    //设置该列数字内容
                    td.innerHTML=intArray[h*15+l];
                    //设置该列对象的背景色
                    td.style.backgroundColor="#f56848";
                    //设置该列对象数字的颜色
                    td.style.color="#000000";
                    //设置对象数字的宽度
                    td.width="30";
                }
            }
};
</script>
</head>
<body>
<h2 style="text-shadow:0.1em 3px 6px blue">从随机生成的数字中抽取 5 的倍数并显示示例</h2>
<table id="table">
</table>
</body>
```

02 为了实现后台线程，需要编写后台执行的 JavaScript 脚本文件，其代码格式如下（详见代码包中的“素材\ch14\14.1.js”）：

```
onmessage = function(event) {
    var data = event.data;
    var returnStr; //将 5 的倍数组成字符串并返回
    var intArray=data.split(";"); //设置返回字符串中数字分隔符为“;”号
    returnStr="";
    for(var i=0;i<intArray.length;i++)
    {
        if(parseInt(intArray[i])%5==0)    //判断能否被 5 整除
        {
            if(returnStr!="")
                returnStr+=";";
            returnStr+=intArray[i];
```

```
            }
        }
        postMessage(returnStr); //返回 5 的倍数组成的字符串
}
```

03 使用 Firefox 浏览器打开编辑好的网页文件，显示效果如图 14-2 所示。

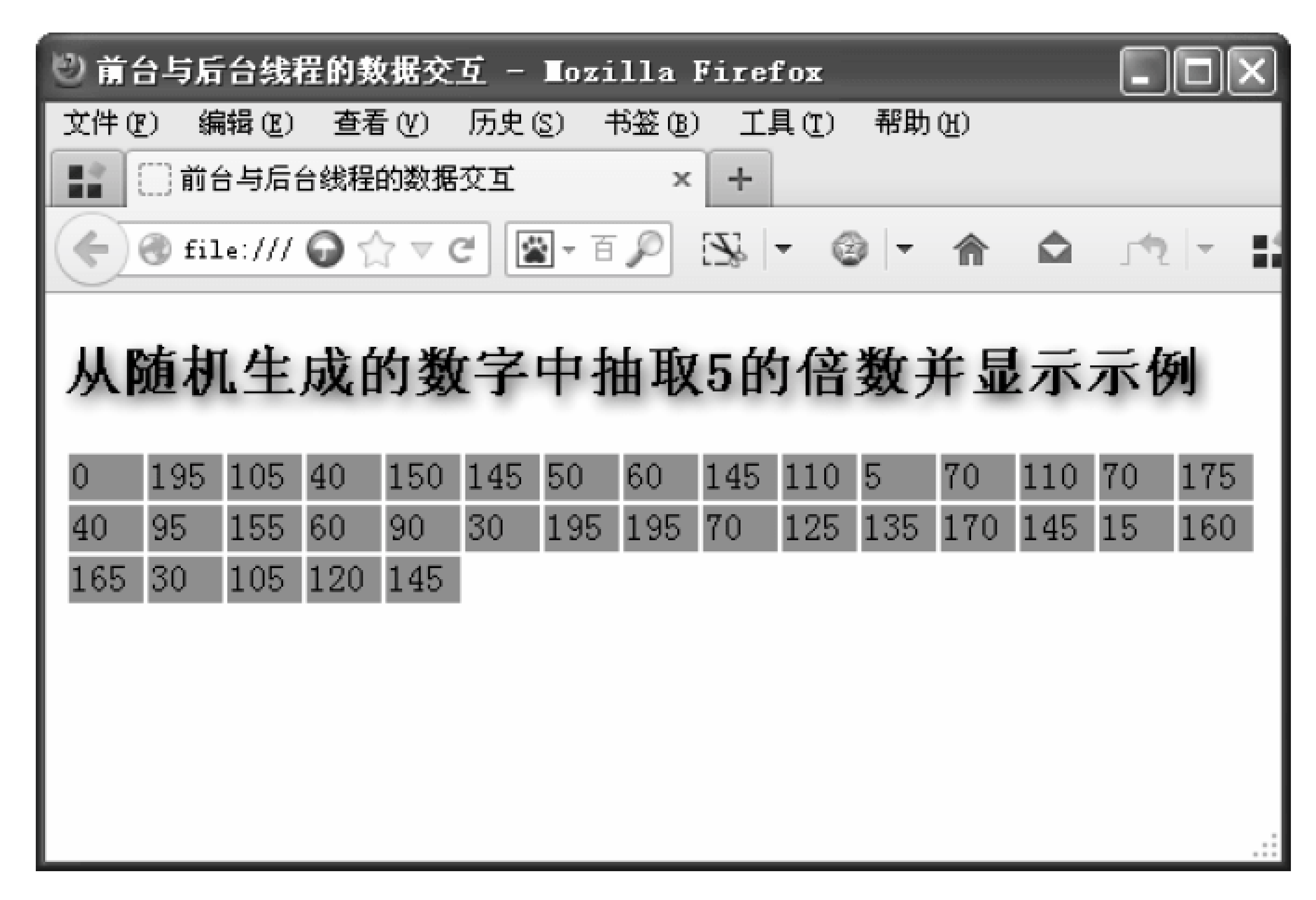


图 14-2 程序运行结果


注 意

由于数字是随机产生的，所以每次生成的数据序列都是不同的。

14.2 线程嵌套

线程中可以嵌套子线程，这样就可以将后台中较大的线程切割成多个子线程，每个子线程独立完成一份工作，可以提高程序的效率。

14.2.1 单线程嵌套

最简单的线程嵌套是单层的嵌套，下面来介绍一个单线程的嵌套案例，该案例所实现的效果和 14.1.3 节中案例的效果相似。其操作方法如下：

01 完成网页前台页面的代码内容，具体代码如下（详见代码包中的“素材\ch14\14.2.html”）：

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<script type="text/JavaScript">
var worker = new Worker("14.2.js");
```

```
worker.postMessage("");
// 从线程中取得计算结果
worker.onmessage = function(event) {
    if(event.data!="")
    {
        var j;    //行号
        var k;   //列号
        var tr;
        var td;
        var intArray=event.data.split(";");
        var table=document.getElementById("table");
        for(var i=0;i<intArray.length;i++)
        {
            j=parseInt(i/10,0);
            k=i%10;
            if(k==0)     //该行不存在
            {
                //添加行
                tr=document.createElement("tr");
                tr.id="tr"+j;
                table.appendChild(tr);
            }
            else   //该行已存在
            {
                //获取该行
                tr=document.getElementById("tr"+j);
            }
            //添加列
            td=document.createElement("td");
            tr.appendChild(td);
            //设置该列内容
            td.innerHTML=intArray[j*10+k];
            //设置该列背景色
            td.style.backgroundColor="blue";
            //设置该列字体颜色
            td.style.color="white";
            //设置列宽
            td.width="30";
        }
    }
};
</script>
```

```
</head>
<body>
<h2 style="text-shadow:0.1em 3px 6px blue">从随机生成的数字中抽取 5 的倍数并显示示例</h2>
<table id="table">
</table>
</body>
```

02 下面编写程序后台执行的主线程的代码内容，该线程用于执行数据挑选，会在 0~200 之间随机产生 200 个随机整数（数字可重复），并将其交与子线程，让子线程挑选可以被 5 整除的数字（详见代码包中的“素材\ch14\14.2.js”）：

```
onmessage=function(event){
    var intArray=new Array(200);      //产生随机的数组
    //生成 200 个随机数
    for(var i=0;i<200;i++)            //数字范围 0-200
        intArray[i]=parseInt(Math.random()*200);
    var worker;
    //调用子线程
    worker=new Worker("14.2-2.js");
    //将随机数组提交给子线程
    worker.postMessage(JSON.stringify(intArray));
    worker.onmessage = function(event) {
        //将挑选结果返回主页面
        postMessage(event.data);
    }
}
```

03 经过步骤 2 中主线程的数字挑选后，可以通过以下子线程将这些数字拼接成字符串，并返回主线程，其操作代码如下（详见代码包中的“素材\ch14\14.2-2.js”）：

```
onmessage = function(event) {
    var intArray= JSON.parse(event.data);
    var returnStr;
    returnStr="";
    for(var i=0;i<intArray.length;i++)
    {
        //判断数字能否被 5 整除
        if(parseInt(intArray[i])%5==0)
        {
            if(returnStr!="")
                returnStr+=";";
            //将所有可以被 5 整除的数字拼接成字符串
            returnStr+=intArray[i];
```

```
            }
        }
        //返回拼接后的字符串至主线程
        postMessage(returnStr);
        //关闭子线程
        close();
    }
```

04 使用 Firefox 浏览器查看网页前台页面，随机产生了一些可以被 5 整除的数字，如图 14-3 所示。



图 14-3　程序运行结果

14.2.2　多个子线程中的数据交互

在实现上述案例时，也可以将子线程再次拆分，生成多个子线程，由多个子线程同时完成工作，这样可以提高处理速度，对较大的 JavaScript 脚本程序来说很有用。

下面将上述案例的程序改为多个子线程嵌套的数据交互案例。

网页前台文件不需要修改，主线程的脚本文件应当做如下修改（详见代码包中的“素材\ch14\14.3.js”）：

```
onmessage=function(event){
    var worker;
    //调用发送数据的子线程
    worker=new Worker("14.3-2.js");
    worker.postMessage("");
    worker.onmessage = function(event) {
        //接收子线程中数据，本示例中为创建好的随机数组
        var data=event.data;
        //创建接收数据子线程
        worker=new Worker("14.2-2.js");
      //把从发送数据子线程中发回消息传递给接收数据的子线程
```

```
            worker.postMessage(data);
        worker.onmessage = function(event) {
                    //获取接收数据子线程中传回数据，本示例中为挑选结果
        var data=event.data;
                    //把挑选结果发送回主页面
                    postMessage(data);
    }
        }
    }
```

上述代码的主线程脚本中提到了两个子线程脚本，其中一个 14.3-2.js 负责创建随机数组，并发送给主线程，另一个 14.2-2.js 负责从主线程接收选好的数组，并进行处理。14.2-2.js 脚本延用 14.2 节脚本文件。14.3-2.js 脚本文件的详细代码如下（详见代码包中的“素材\ch14\14.3-2.js”）：

```
onmessage = function(event) {
var intArray=new Array(200);
for(var i=0;i<200;i++)
    intArray[i]=parseInt(Math.random()*200);
postMessage(JSON.stringify(intArray));
close();
}
```

执行后的效果如图 14-4 所示。



图 14-4　程序运行结果



提 示

通过以上几个案例的展示，其最终显示结构都是相同的，只是代码的编辑与线程的嵌套有所差异，在实际的应用中合理地嵌套子线程虽然会使代码结构变复杂，但是却能在很大程度上提高程序的处理效率。

14.3 综合实例——创建 Web Worker 计数器

本实例主要创建了简单的 Web Worker，实现在后台计数的功能。具体操作步骤如下：

01 首先创建外部的 JavaScript 文件"workers01.js"，主要用于计数。代码如下：

```
var i=0;

function timedCount()
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",500);
}
timedCount();
```

02 以上代码中重要的部分是 postMessage()方法，主要用于向 HTML 页面传回一段消息。下面创建 HTML 页面的代码如下：

```
<!DOCTYPE html>
<html>
<body>
<p>计数:<output id="result"></output></p>
<button onclick="startWorker()">开始  Worker</button>
<button onclick="stopWorker()">停止  Worker</button>
<br /><br />
<script>
var w;
function startWorker()
{
<!—首先判断浏览器是否支持 Web Worker -->
if(typeof(Worker)!=="undefined")
    {
<!—检测是否存在 worke，如果不存在，它会创建一个新的 Web Worker 对象，然后运行 " workers.js01"
中的代码-->
    if(typeof(w)=="undefined")
    {
    w=new Worker("/workers01.js");
   }
<!—向 Web Worker 添加一个"onmessage"事件监听器-->
    w.onmessage = function (event) {
     document.getElementById("result").innerHTML=event.data;
     };
```

```
    }
else
    {
    document.getElementById("result").innerHTML="对不起，您的浏览器不支持 Web Workers...";
    }
}
function stopWorker()
{
<!—终止 Web Worker，并释放浏览器/计算机资源-->
w.terminate();
}
</script>
</body>
</html>
```

运行结果如图 14-5 所示。

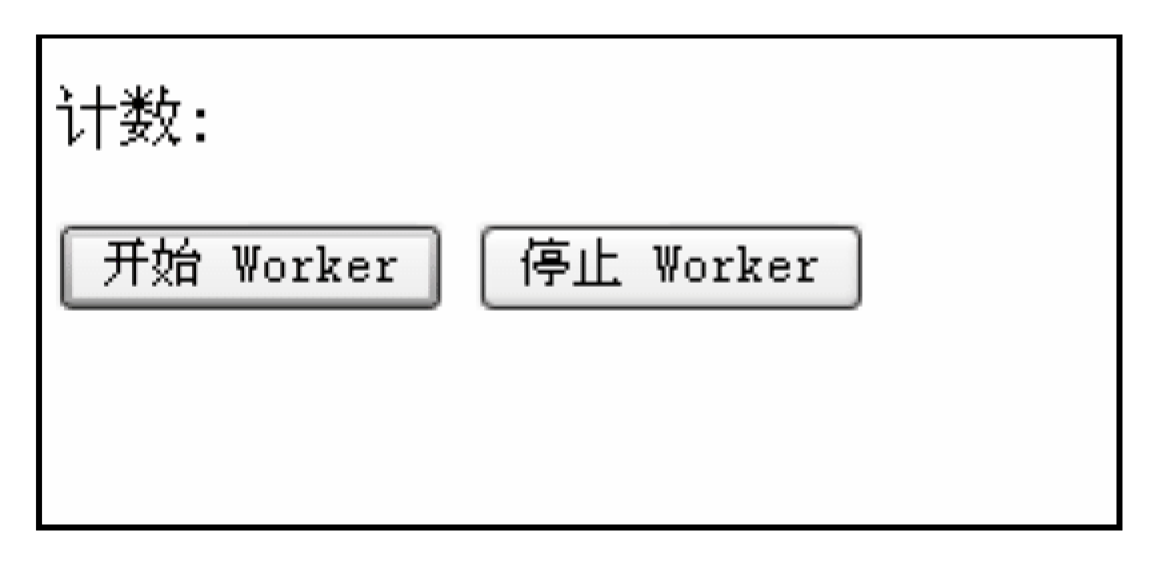


图 14-5　程序运行结果

14.4　问题解答

1．工作线程（Web Worker）的主要应用场景有哪些？

工作线程的主要应用场景有三个，分别如下：

- 使用工作线程做后台数值（算法）计算。
- 使用共享线程处理多用户并发连接。
- HTML 5 线程代理。

2．目前浏览器对 Web Worker 的支持情况如何？

目前大部分主流的浏览器都支持 Web Worker，但是 IE 目前并不支持。

第 15 章　构建离线 Web 应用程序

网页离线应用程序是实现离线 Web 应用的重要技术。目前已有的离线 Web 应用程序很多，都需要本书知识做基础。通过本章的学习，读者能够掌握 HTML 5 离线应用程序的基础知识，了解离线应用程序的实现方法。

15.1　HTML 5 离线应用程序

为了能在离线的情况下访问网站，可以采用 HTML 5 的离线 Web 功能。下面来学习 Web 应用程序如何缓存。

15.1.1　本地缓存

在 HTML 5 中，新增了本地缓存，也就是 HTML 离线 Web 应用，主要是通过应用程序缓存整个离线网站的 HTML、CSS、JavaScript、网站图像和资源。当服务器没有和 Internet 建立连接的时候，也可以利用本地缓存中的资源文件来正常运行 Web 应用程序。

另外如果网站发生了变化，应用程序缓存将重新加载变化的数据文件。

15.1.2　本地缓存与浏览器网页缓存的区别

浏览器网页缓存与本地缓存的主要区别如下：

- 浏览器网页缓存主要是为了加快网页加载的速度，所以会对每一个打开的网页都进行缓存操作，而本地缓存是为整个 Web 应用程序服务的，只缓存那些指定缓存的网页。
- 在网络连接的情况下，浏览器网页缓存一个页面的所有文件，但是一旦离线，用户单击链接时，将会得到一个错误消息。而本地缓存在离线时，仍然可以正常访问。
- 对于网页浏览者而言，浏览器网页缓存了哪些内容和资源，这些内容是否安全等都未知；而本地缓存的页面是编程人员指定的内容，所以相对比较安全。

15.1.3　支持离线行为

要支持离线行为，首先要能够判断网络连接状态，在 HTML 5 中引入了一些判断应用程序网络连接是否正常的新的事件。对应应用程序的在线状态和离线状态会有不同的行为模式。

用于实现在线状态监测的是 window.navigator 对象的属性。其中的 navigator.online 属性是一个标明浏览器是否处于在线状态的布尔属性，当 online 值为 true 并不能保证 web 应用程序在用户的机器上一定能访问到相应的服务器，而当其值为 false 时，不管浏览器是否真正联网，

应用程序都不会尝试进行网络连接。

监测页面状态是在线还是离线的具体代码如下：

```
//页面加载的时候，设置状态为 online 或 offline
function loaddemo(){
  if (navigator.online) {
    log(“online”);
} else {
  log(“offline”);
}
}
//添加事件监听器，在线状态发生变化时，触发相应动作
window.addeventlistener(“online”,function€{
}，true);
window.addeventlistener(“offline”,function(e) {
  log(“offline”);
},true);
```



上述代码可以在 IE 浏览器中使用。

15.2 了解 Manifest（清单）文件

那么客户端的浏览器是如何分清应该缓存那些文件？这就需要依靠 manifest 文件来管理。manifest 文件是一个简单的文本文件，在该文件中以清单的形式列举了需要被缓存或不需要被缓存的资源文件的文件名称以及这些资源文件的访问路径。

Manifest 文件把指定的资源文件类型分为三类，分别是 CACHE、NETWORK 和 FALLBACK，这三类的含义分别如下。

- CACHE 类别：该类别指定需要被缓存在本地的资源文件。这里需要特别注意的是：如果为某个页面指定需要本地缓存的资源文件时，不需要把这个页面本身指定在 CACHE 类型中，因为如果一个页面具有 manifest 文件，浏览器会自动对这个页面进行本地缓存。
- NETWORK 类别：该类别为不进行本地缓存的资源文件，只有当客户端与服务器端建立连接的时候才能访问这些资源文件。
- FALLBACK 类别：该类别中指定两个资源文件，其中一个资源文件为能够在线访问时使用的资源文件，另一个资源文件为不能在线访问时使用的备用资源文件。

以下是一个简单的 manifest 文件的内容：

```
CACHE MANIFEST
```

```
#文件的开头必须是 CACHE MANIFEST
CACHE:
123.html
myphoto.jpg
12.php
NETWORK:
http://www.baidu.com/xxx
feifei.php
FALLBACK:
online.js locale.js
```

上述代码含义分析如下：

- 指定资源文件，文件路径可以是相对路径，也可以是绝对路径。指定时每个资源文件为独立的一行。
- 第一行必须是 CACHE MANIFEST，此行的作用告诉浏览器需要对本地缓存中的资源文件进行具体设置。
- 每类型都必须出现，而且同一个类别可以重复出现。如果文件开头没有指定类别而直接书写资源文件的时候，浏览器把这些资源文件视为 CACHE 类别。
- 在 manifest 文件中，注释行以“#”开始，主要作用是进行一些必要的说明或解释。

为单个网页添加 manifest 文件时，需要在 Web 应用程序页面上的 html 元素的 manifest 属性中指定 manifest 文件的 URL 地址。具体的代码如下：

```
<html manifest="123.manifest">
</html>
```

添加上述代码后，浏览器就能够正常地阅读该文本文件了。



用户可以为每个页面单独指定一个 mainifest 文件，也可以对整个 Web 应用程序指定总的 manifest 文件。

上述操作完成后，即可实现资源文件缓存到本地的目的。当要对本地缓存区的内容进行修改时，只需修改 manifest 文件。文件被修改后，浏览器可以自动检查 manifest 文件，并自动更新本地缓存区中的内容。

15.3 了解 applicationcache API

传统的 Web 程序中浏览器也会对资源文件进行 cache，但是并不是很可靠，有时起不到预期的效果。而 HTML 5 中的 application cache 支持离线资源的访问，为离线 Web 应用的开发提供了可能。

使用 application cache API 的好处有以下几点。

- 用户可以在离线时继续使用；
- 缓存到本地，节省带宽，加速用户体验的反馈；
- 减轻服务器的负载。

Applicationcache API 是一个操作应用缓存的接口，是 windows 对象的直接子对象 window.applicationcache。window.applicationcache 对象可触发一系列与缓存状态相关的事件。具体事件如下表所示。

事件	接口	触发条件	后续事件
checking	Event	用户代理检查更新或者在第一次尝试下载 manifest 文件的时候，本事件往往是事件队列中第一个被触发的	noupdate, downloading, obsolete, error
noupdate	Event	检测出 manifest 文件没有更新	无
downloading	Event	用户代理发现更新并且正在获取资源，或者第一次下载 manifest 文件列表中列举的资源	progress, error, cached, updateready
progress	ProgressEvent	用户代理正在下载 manifest 文件中需要缓存的资源	progress, error, cached, updateready
cached	Event	manifest 中列举的资源已经下载完成，并且已经缓存	无
updateready	Event	manifest 中列举的文件已经重新下载并更新成功，接下来 JavaScript 可以使用 swapCache()方法更新到应用程序中	无
obsolete	Event	manifest 的请求出现 404 或者 410 错误，应用程序缓存被取消	无

此外，没有可用更新或者发生错误时，还有一些表示更新状态的事件：

```
Onerror
Onnoupdate
onprogress
```

该对象有一个数值型属性 window.applicationcache.status，代表了缓存的状态。缓存状态共有 6 种，如下表所示。

数值型属性	缓存状态	含义
0	UNCACHED	未缓存
1	IDLE	空闲
2	CHECKING	检查中

（续表）

数值型属性	缓存状态	含义
3	DOWNLOADING	下载中
4	UPDATEREADY	更新就绪
5	OBSOLETE	过期

window.applicationcache 有三个方法，如下表所示。

方法名	描述
update()	发起应用程序缓存下载进程
abort()	取消正在进行的缓存下载
swapcache()	切换成本地最新的缓存环境



调用 update()方法会请求浏览器更新缓存。包括检查新版本的 manifest 文件并下载必要的新资源。如果没有缓存或者缓存已过期，则会抛出错误。

15.4 浏览器对 Web 离线应用的支持情况

不用的浏览器版本对 Web 离线应用技术的支持情况是不同的，下表是常见浏览器对 Web 离线应用的支持情况。

浏览器名称	支持 Web 离线应用的浏览器版本
IE	IE 9 及更低版本目前尚不支持
Firefox	Firefox 3.5 及更高版本
Opera	Opera 10.6 及更高版本
Safari	Safari 4 及更高版本
Chrome	Chrome 5 及更高版本
Android	Android 2.0 及更高版本

15.5 综合实例——离线定位跟踪

下面结合上述内容的学习来构建一个离线 Web 应用程序，具体内容如下：

01 创建记录资源的 manifest 文件。首先要创建一个缓冲清单文件 123.manifest，文件中列出了应用程序需要缓存的资源。具体实现代码如下：

```
CACHE MANIFEST
# JavaScript
./offline.js
#./123.js
```

```
./log.js
#stylesheets
./CSS.css
#images
```

02 创建构成界面的 HTML 和 CSS。下面来实现网页结构，其中需要指明程序中用到的 JavaScript 文件和 CSS 文件，并且还要调用 manifest 文件。具体实现代码如下：

```
<!DOCTYPE html >
<html lang="en" manifest="123.manifest">
<head>
<title>创建构成界面的 html 和 CSS</title>
<script src="log.js"></script>
<script src="offline.js"></script>
<script src="123.js"></script>
<link rel="stylesheet" href="CSS.css" />
</head>
<body>
        <header>
    <h1>Web 离线应用</h1>
    </header>
    <section>
    <article>
        <button id="installbutton">check for updates</button>
        <h3>log</h3>
        <div id="info">
        </div>
        </article>
     </section>
</body>
</html>
```



上述代码中有两点需要注意。其一，因为使用了 manifest 特性，所以 HTML 元素不能省略（为了使代码简洁，HTML 5 中允许省略不必要的 HTML 元素）。其二，代码中引入了按钮，其功能是允许用户手动安装 Web 应用程序，以支持离线情况。

03 创建离线的 JavaScript。在网页设计中经常会用到 JavaScript 文件，该文件通过<script>标签引入网页。在执行离线 Web 应用时，这些 JavaScript 文件也会一并存储到缓存中。

```
<offline.js>
/*
 *记录 window.applicationcache 触发的每一个事件
```

```
 */
window.applicationcache.onchecking =
function(e) {
        log("checking for application update");
       }
window.applicationcache.onupdateready =
function(e) {
        log("application update ready");
       }
window.applicationcache.onobsolete =
function(e) {
        log("application obsolete");
       }
window.applicationcache.onnoupdate =
function(e) {
        log("no application update found");
       }
window.applicationcache.oncached =
function(e) {
        log("application cached");
       }
window.applicationcache.ondownloading =
function(e) {
        log("downloading application update");
       }
window.applicationcache.onerror =
function(e) {
        log("online");
       }, true);
/*
 *将 applicationcache 状态代码转换成消息
 */
 showcachestatus = function(n) {
      statusmessages = ["uncached","idle","checking","downloading","update ready","obsolete"];
      return statusmessages[n];
}
install = function(){
        log("checking for updates");
      try {
        window.applicationcache.update();
       } catch (e) {
               applicationcache.onerror();
```

```
        }
    }
onload = function(e) {
        //检测所需功能的浏览器支持情况
       if(!window.applicationcache) {
        log("html 5 offline applications are not supported in your browser.");
            return;
        }
        if(!window.localstorage) {
        log("html 5 local storage not supported in your browser.");
            return;
        }
        if(!navigator.geolocation) {
        log("html 5 geolocation is not supported in your browser.");
            return;
        }
        log("initial cache status: " + showcachestatus(window.applicationcache.status));
        document.getelementbyid("installbutton").onclick = checkfor;
}
<log.js>
log = function() {
        var p = document.createelement("p");
        var message = array.prototype.join.call(arguments," ");
       p.innerhtml = message
       document.getelementbyid("info").appendchild(p);
}
```

04 检查 applicationCache 的支持情况。并非所有浏览器都可以支持 applicationCache 对象，所以在编辑时需要加入浏览器支持性检测功能，并提醒浏览者页面无法访问是浏览器兼容问题。具体实现代码如下：

```
onload = function(e) {
   // 检测所需功能的浏览器支持情况
   if (!window.applicationcache) {
      log("您的浏览器不支持 html 5 Offline Applications ");
      return;
   }
   if (!window.localStorage) {
      log("您的浏览器不支持 html 5 Local Storage    ");
      return;
   }
if (!window.WebSocket) {
      log("您的浏览器不支持 html 5 WebSocket ");
```

```
        return;
    }
  if (!navigator.geolocation) {
        log("您的浏览器不支持 html 5 Geolocation ");
        return;
    }
        log("lnitial cache status:" + showCachestatus(window.applicationcache.status));
    document.getelementbyld("installbutton").onclick = install;
  }
```

05 为 update 按钮添加处理函数。下面来设置 update 按钮的行为函数，该函数功能为执行更新应用缓存，具体代码如下：

```
install = function() {
log("checking for updates");
try {
window.applicationcache.update();
} catch (e) {
applicationcache.onerror():
}
}
```


注 意

单击按钮后将检查缓存区，并更新需要更新的缓存资源。当所有可用更新都下载完毕之后，将向用户界面返回一条应用程序安装成功的提示信息，接下来用户就可以在离线模式下运行了。

06 添加 Storage 功能代码。当应用程序处于离线状态时，需要将数据更新写入本地存储，本实例使用 storage 实现该功能，因为当上传请求失败后可以通过 storage 得到恢复。如果应用程序遇到某种原因导致网络错误，或者应用程序被关闭的时候，数据会被存储以便下次再进行传输。

实现 storage 功能的具体代码如下：

```
var storelocation =function(latitude, longitude){
//加载 localstorage 的位置列表
var locations = json.pares(localstorage.locations || "[]");
//添加地理位置数据
locations.push({"latitude" : latitude, "longitude" : longitude});
//保存新的位置列表
localstorage。Locations = json.stringify(locations);
```

由于 localstorage 可以将数据存储在本地浏览器中，特别适用于具有离线功能的应用程序，所以本实例中用它来保存坐标。本地存储中的缓存数据在网络连接恢复正常后，应用程序会自

动与远程服务器进行数据同步。

07 添加离线事件处理程序。对于离线 Web 应用程序，在使用时要结合当前状态执行特定的事件处理程序，本实例中的离线事件处理程序设计如下:

- 如果应用程序在线，事件处理函数会存储并上传当前坐标;
- 如果应用程序离线，事件处理函数只存储不上传;
- 当应用程序重新连接到网络后，事件处理函数会在 UI 上显示在线状态，并在后台上传之前存储的所有数据。

具体实现代码如下：

```
window.addeventlistener(“online”, function(e){
    Log(“online”);
}, true);
window.addeventlistener(“offline”, function(e) {
   Log(“offline”);
}, true);
```

网络连接状态在应用程序没有真正运行的时候可能会发生改变，例如用户关闭了浏览器，刷新页面或跳转到了其他网站，为了应对这些情况，离线应用程序在每次页面加载时都会检查与服务器的连接状况，如果正常，会尝试与远程服务器同步数据：

```
if(navigator.online){
   Uploadlocations();
}
```

在 IE 9.0 浏览器中的预览效果如图 15-1 所示。



图 15-1 程序运行结果

15.6 问题解答

1. 如何更新离线存储？

更新 HTML 5 离线缓存主要有三种方法：

- 用户清除了离线存储的数据，这样不一定是清理浏览器历史记录就能做到的，因为不同浏览器管理离线存储的方式不同。比如 Firefox 的离线存储数据要到“选项”→“高级”→“网络”→“脱机存储”里才可以清除。
- 修改 manifest 文件。
- 使用 JavaScript API 编写更新程序。

2. 离线存储功能的主要作用有哪些？

目前，在最新的主流浏览器中都已添加了对 HTML 5 的 offline storage 功能的支持，HTML 5 离线存储功能非常强大，它的作用是：在用户没有与因特网连接时，照样可以访问站点或应用，在用户与因特网连接时，自动更新缓存数据。

第 16 章　HTML 5 的拖放功能

拖放是一种常见的特性，即抓取对象以后拖到另一个位置，在 HTML 5 中，拖放是标准的一部分，任何元素都能够被拖放。常见的拖放元素为图片、文字等。

16.1　一个简单的拖放实例

下面给出一个简单的拖放实例，主要实现的功能就是把一张图片拖放到一个矩形当中，实例的具体实现代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
#div1 {width:150px;height:150px;padding:10px;border:1px solid #aaaaaa;}
</style>
<script type="text/JavaScript">
function allowDrop(ev)
{
ev.preventDefault();
}
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
<p>请把图片拖放到矩形中：</p>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
```

```
<br />
<img id="drag1" src="images/2.gif" draggable="true" ondragstart="drag(event)" />
</body>
</html>
```

将上述代码保存为.html 格式，在 IE 浏览器中预览效果，如图 16-1 所示。



图 16-1 预览效果

可以看到当选中图片后，在不松开鼠标的情况下，可以将其拖放到矩形框中，如图 16-2 所示。



图 16-2 拖放图片

代码解释如下：

- 调用 preventDefault()来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）。
- 通过 dataTransfer.getData("Text")方法获得被拖的数据。该方法将返回在 setData()方法中设置为相同类型的任何数据。
- 被拖数据是被拖元素的 id("drag1")。

● 把被拖元素追加到放置元素（目标元素）中。

16.2 分析拖放的实现过程

HTML 实现拖放效果，常用的实现方法是利用 HTML 5 新增加的事件 drag 和 drop，下面就具体分析拖放实现的过程。

1. 设置元素为可拖放

首先，为了使元素可拖动，把 draggable 属性设置为 true，具体的代码如下：

```
<img draggable="true" />
```

2. 拖动内容

实现拖放的第二步就是设置拖动的元素，常见的元素有图片、文字、动画等。实现拖放功能的是 ondragstart 和 setData()，即规定当元素被拖动时，会发生什么。

例如：在上面的例子中，ondragstart 属性调用了一个函数 drag（event），它规定了被拖动的数据。

dataTransfer.setData()方法设置被拖数据的数据类型和值，具体的代码如下：

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

在这个例子中，数据类型是"Text"，值是可拖动元素的 id("drag1")。

3. 放置位置

实现拖放功能的第三步就是将可拖放元素放到适合位置，实现该功能的事件是 ondragover，默认情况下，无法将数据/元素放置到其他元素中。如果需要设置允许放置，用户必须阻止对元素的默认处理方式。

这就需要通过调用 ondragover 事件的 event.preventDefault()方法，具体的代码如下：

```
event.preventDefault()
```

4. 进行放置

当放置被拖元素时，就会发生 drop 事件。在上面的例子中，ondrop 属性调用了一个函数 drop（event），其具体的代码如下：

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
```

```
ev.target.appendChild(document.getElementById(data));
}
```

16.3 浏览器对拖放功能的支持情况

不用的浏览器版本对拖放功能的支持情况是不同的，下表是常见浏览器对拖放功能的支持情况。

浏览器名称	支持拖放功能的浏览器版本
IE	IE 9.0 及更高版本
Firefox	Firefox 3.6 及更高版本
Opera	Opera 12.0 及更高版本
Safari	Safari 5 及更高版本
Chrome	Chrome 5 及更高版本

16.4 综合实例 1——在网页中拖放文字

在了解了 HTML 5 的拖放技术后，下面给出一个具体的实例，实现的效果就是在网页中拖放文字。

具体的代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
<title>拖放文字</title>
<style>
body {
    font-family: 'Microsoft YaHei';
}
div.drag {
    background-color:#AACCFF;
    border:1px solid #666666;
    cursor:move;
    height:100px;
    width:100px;
    margin:10px;
    float:left;
}
div.drop {
    background-color:#EEEEEE;
    border:1px solid #666666;
```

```
        cursor: pointer;
        height:150px;
        width:150px;
        margin:10px;
        float:left;
}
</style>
</head>
<body>
<div draggable="true" class="drag"
        ondragstart="dragStartHandler(event)">Drag me!</div>
<div class="drop"
        ondragenter="dragEnterHandler(event)"
        ondragover="dragOverHandler(event)"
        ondrop="dropHandler(event)">Drop here!<ol /></div>
<script>
var internalDNDType = 'text';
function dragStartHandler(event) {
    event.dataTransfer.setData(internalDNDType,
                                            event.target.textContent);
        event.effectAllowed = 'move';
}
// dragEnter 事件
function dragEnterHandler(event) {
    if (event.dataTransfer.types.contains(internalDNDType))
        if (event.preventDefault) event.preventDefault();}
// dragOver 事件
function dragOverHandler(event) {
        event.dataTransfer.dropEffect = 'copy';
        if (event.preventDefault) event.preventDefault();
}
function dropHandler(event) {
        var data = event.dataTransfer.getData(internalDNDType);
        var li = document.createElement('li');
        li.textContent = data;
        event.target.lastChild.appendChild(li);
}
</script>
</body>
</html>
```

下面介绍实现拖放的操作步骤。

01 将上述代码保存为.html 格式的文件，在 IE 9.0 浏览器的预览效果如图 16-3 所示。



图 16-3 预览效果

02 选中左边矩形中的元素，将其拖曳到右边的矩形放开，如图 16-4 所示。

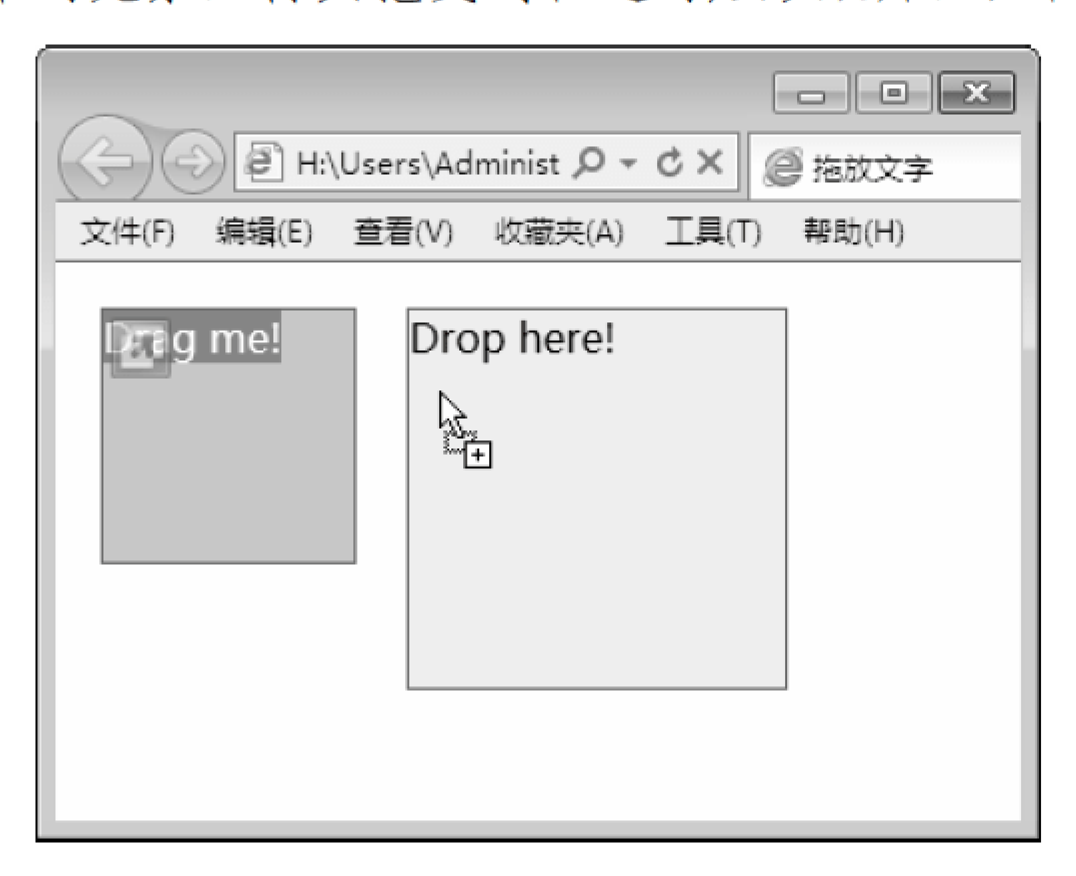


图 16-4 选择被拖放文字

03 松开鼠标，可以看到拖放之后的效果，如图 16-5 所示。



图 16-5 拖放一次的效果

04 还可以多次拖放文字元素，效果如图 16-6 所示。



图 16-6 拖放多次的效果

16.5 综合实例 2——在网页中来回拖放图片

下面再给出一个具体的实例，实现的效果就是在网页中来回拖放图片。

具体的代码如下：

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
#div1, #div2
{float:left; width:100px; height:35px; margin:10px;padding:10px;border:1px solid #aaaaaa;}
</style>
<script type="text/JavaScript">
function allowDrop(ev)
{
ev.preventDefault();
}
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
```

```
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
  <img src="images/1.gif" draggable="true" ondragstart="drag(event)" id="drag1" />
</div>
<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
</body>
</html>
```

在记事本中输入这些代码，然后将其保存为.html 格式，使用 IE 9.0 浏览器查看该文件，即可在页面中看到效果，选中网页中的图片，即可在两个矩形当中来回拖放，如图 16-3 所示。

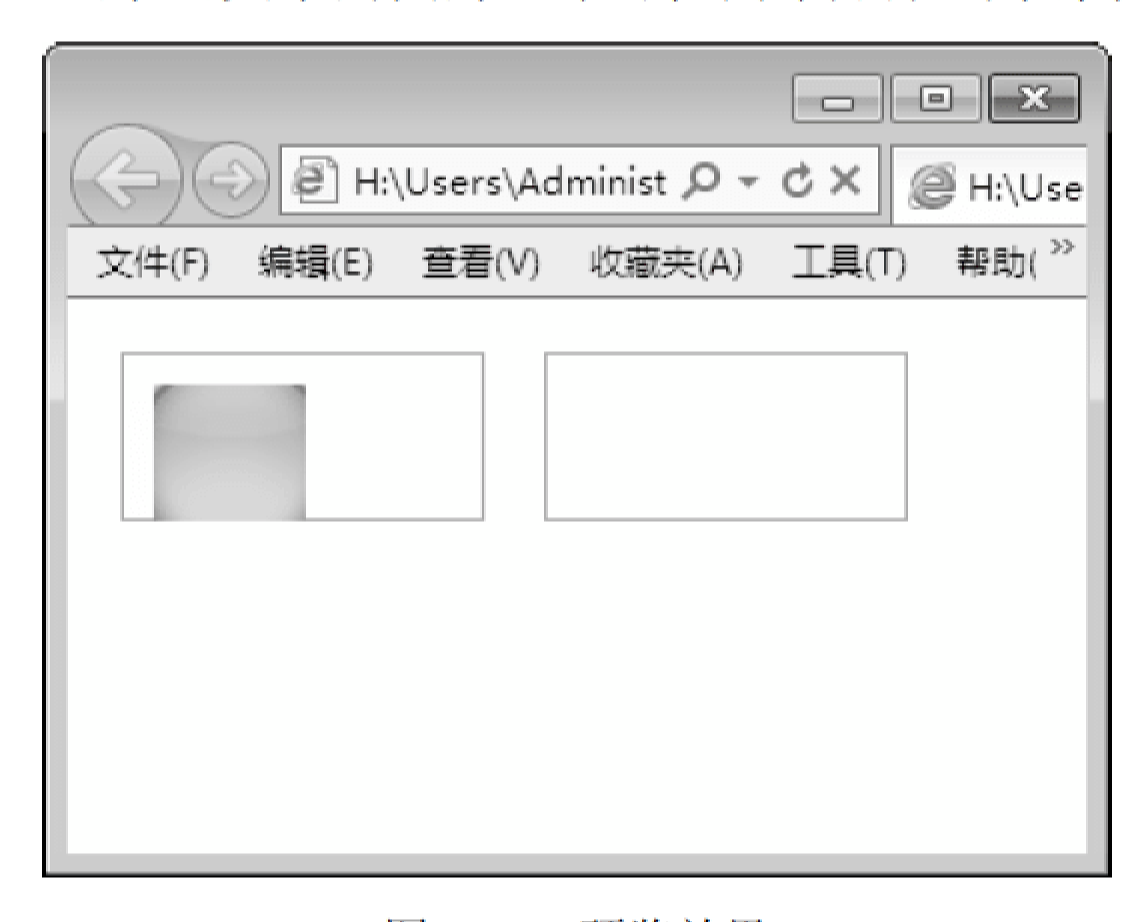


图 16-7　预览效果

16.6　问题解答

1．在 HTML 5 中，实现拖放效果的方法唯一吗？

在 HTML 5 中，实现拖放效果的方法并不是唯一的。除了可以使用事件 drag 和 drop 外，还可以使用利用 canvas 标签来实现。

2．在 HTML 5 中，可拖放的对象只有文字和图像吗？

默认情况下，图像、链接和文本是可以拖动的，也就是说，不用额外编写代码，用户就可以拖动它们。文本只有在被选中的情况下才能拖动，而图像和链接在任何时候都可以拖动。

也可以让其他元素拖动，HTML 5 为所有 HTML 元素规定了一个 draggable 属性，表示元素是否可以拖动。图像和链接的 draggable 属性自动被设置成了 true，而其他元素这个属性的默认值都是 false。要想让其他元素可拖动，或者让图像或链接不能拖动，都可以设置该属性。

第 17 章　HTML 5 服务器发送事件

HTML 5 规范定义了 Server-Sent Event 和 Web Socket 两个特性，为浏览器变成一个 RIA（Rich Internet Applications 富互联网应用）客户端平台提供了强大的支持，使用这两个特性，可以帮助服务器将数据“推送”到客户端浏览器。本章节主要讲述服务器发送事件的基本概念、服务器发送事件的实现过程等。

17.1　服务器发送事件概述

在网页客户端更新过程中，如果使用早期技术，网页不得不询问是否有可用的更新，这样将不能很好地实时地获取服务器的信息，并且加大了资源的耗费。在 HTML 5 中，通过服务器发送事件，可以让网页客户端自动获取来自服务器的更新。

服务器发送事件（Server-Sent Event）允许网页获得来自服务器的更新，这种数据的传递和前面讲述的 Web Socket 不同。服务器发送事件是单向传递信息，服务器将更新的信息自动发送到客户端，而 Web Socket 是双向通信技术。

目前，常见浏览器对 Server-Sent Event 的支持情况如下表所示。

浏览器名称	支持 Server-Sent Event 的浏览器版本
IE	不支持
Firefox	Firefox 3.6 及更高版本
Opera	Opera 12.0 及更高版本
Safari	Safari 5 及更高版本
Chrome	Chrome 5 及更高版本

17.2　服务器发送事件的实现过程

了解完服务器发送事件的基本概念后，下面来学习实现过程。

17.2.1　检测浏览器是否支持 Server-Sent Event

首先可以检查客户端浏览器是否支持 Server-Sent 事件。代码如下：

```
if(typeof(EventSource)!=="undefined")
  {
  // 浏览器支持的情况
```

```
    }
else
  {
  // 对不起，您的浏览器不支持……
  }
```

用户在代码中设置提示信息，这样如果浏览者的客户端不支持，将会显示提示信息。

17.2.2 EventSource 对象

在 HTML 5 中的服务器发送事件中，使用 EventSource 对象接收服务器发送事件的通知。该对象的事件含义如下表所示。

事件名称	含义
onopen	当连接打开时触发该事件
onmessage	当收到信息时触发该事件
onerror	当连接关闭时触发该事件

在事件处理函数中，可以通过使用 readyState 属性检测连接状态，主要有三种状态，如下表所示。

状态名称	值	含义
CONNECTING	0	正在建立连接
OPEN	1	连接已经建立，正在委派事件
CLOSED	2	连接已经关闭

例如下面的代码就是使用了 onmessage 的实例：

```
var source=new EventSource("/123.php");
source.onmessage=function(event)
  {
  document.getElementById("result").innerHTML+=event.data + "<br />";
  };
```

其中，该代码创建一个新的 EventSource 对象，然后规定发送更新页面的 URL（本例中是"/123.php"）。每接收到一次更新，就会发生 onmessage 事件。当 onmessage 事件发生时，把已接收的数据推入 id 为"result" 的元素中。

17.2.3 服务器端代码

为了让上面的例子可以运行，还需要能够发送数据更新的服务器（比如 PHP 和 ASP）。服务器端事件流的语法是非常简单，把"Content-Type"报头设置为"text/event-stream"，就可以开始发送事件流了。

如果服务器是 PHP，则服务器的代码如下：

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');
$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

如果服务器是 ASP，则服务器的代码如下：

```
<%
Response.ContentType="text/event-stream"
Response.Expires=-1
Response.Write("data: " & now())
Response.Flush()
%>
```

上面的代码中，把报头"Content-Type"设置为 "text/event-stream"，规定不对页面进行缓存，输出发送日期（始终以 "data: " 开头），向网页刷新输出数据。

17.3 综合实例——向服务器端发送事件

下面通过一个综合的案例，详细介绍整个代码的操作过程。

01 首先创建运行主页文件，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset=\"UTF-8\">
</head>
<body>
<h1>获得服务器更新</h1>
<div id="result">
</div>
<script>
if(typeof(EventSource)!=="undefined")
  {
  var source=new EventSource("/123.php");
  source.onmessage=function(event)
    {
    document.getElementById("result").innerHTML+=event.data + "<br />";
    };
```

```
    }
  else
    {
    document.getElementById("result").innerHTML="对不起，您的浏览器不支持服务器发送事件...";
    }
</script>
</body>
</html>
```



这里规定通信数据的编码为 UTF-8 格式，另外所有的页面编码都要统一为 UTF-8，否则会乱码或无数据。

02 编写服务器端文件 123.php，代码如下：

```
<?php
error_reporting(E_ALL);
//注意：发送包头定义 MIMIE 类型（header 部分），是实现服务器推送信息必须的代码（MIMIE 类型定义了事件框架格式）
header(\"Content-Type:text/event-stream\");
echo 'data:服务器第一次发送数据'.\"\n\";
echo 'data:服务器第二次发送数据'.\"\n\";
?>
```



输出的格式必须为 data:value 格式，这是 text/event-tream 格式规定。

03 在 IE 浏览器中的访问主页文件效果如图 17-1 所示。

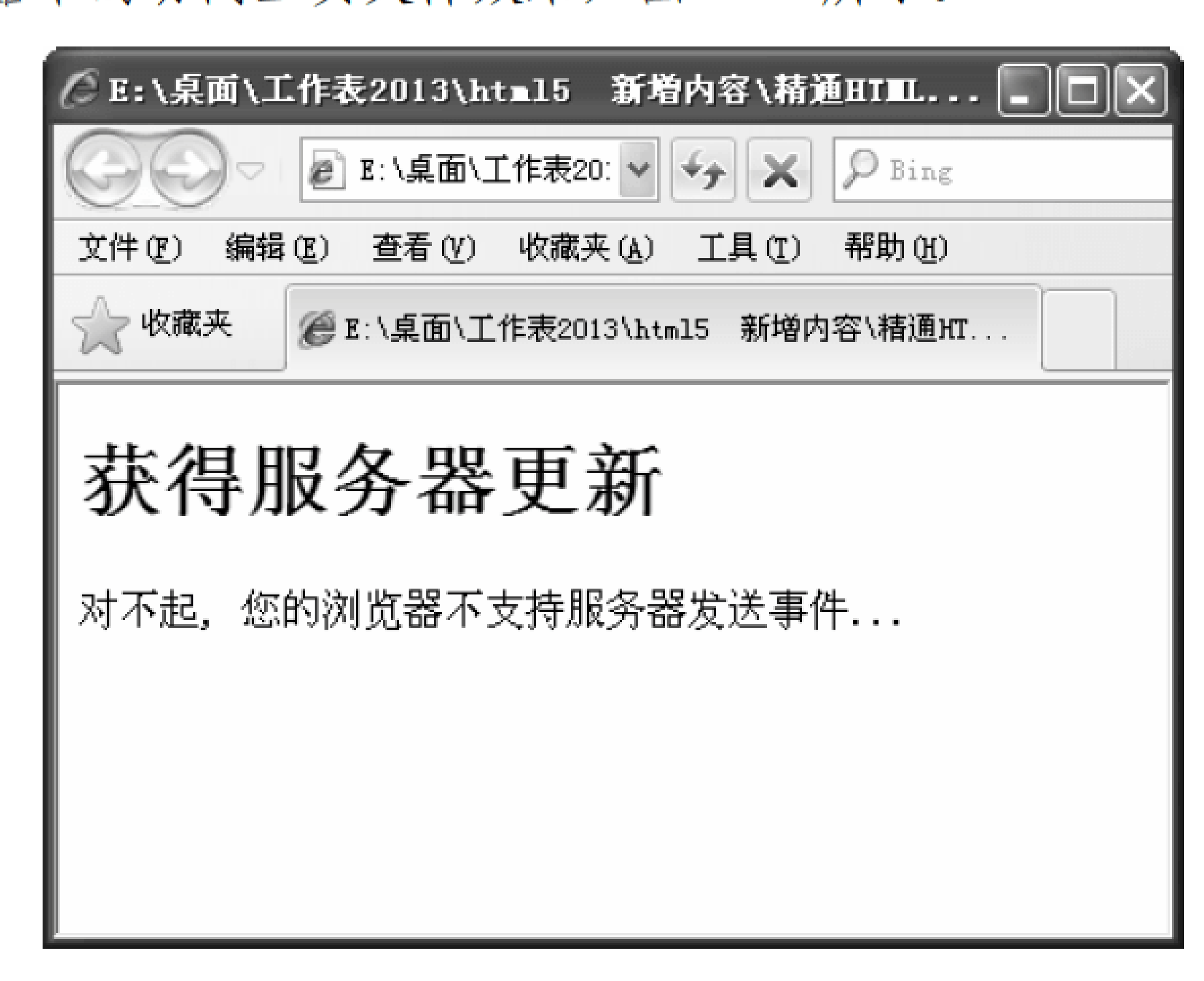


图 17-1 代码运行效果

04 在 Firefox 浏览器中的访问主页文件效果如图 17-2 所示。服务器每隔一段时间推送一个此数据。



图 17-2 代码运行效果

17.4 问题解答

1. 如何编写 JSP 的服务器端代码？

如果服务器端是 JSP，服务器的代码段如下：

```
<%@ page contentType="text/event-stream; charset=UTF-8"%>
<%
    response.setHeader("Cache-Control", "no-cache");
    out.print("data: >>server Time" + new java.util.Date() );
    out.flush();
%>
```

其中，编码要采用统一的 UTF-8 格式。

2. 如何优化服务器端代码？

EventSource 对象是一个不间歇运行的程序，时间一长会大量的消耗资源，甚至导致客户端浏览器崩溃，那么如何优化执行代码呢？

在 HTML 5 中使用 Web Workers 优化 JavaScript 执行复杂运算、重复运算和多线程运算；对于执行时间长、消耗内存多的 JavaScript 程序代码最为有用。Web Workers 的使用方法请参照本书的第 12 章的内容。

第 18 章　HTML 5、CSS3 和 JavaScript 搭配应用

网页吸引人之处，莫过于具有动态效果，利用 CSS 伪类元素可以轻易实现超级链接的动态效果。不过利用 CSS 能实现的动态效果非常有限。在网页设计中，还可以将 CSS 与 JavaScript 结合，创建出具有动态效果的页面。

18.1　综合实例 1——打字效果的文字

文字是网页的灵魂，没有文字的网页，不管特效多么绚丽多彩必定没有任何实际意义。文字特效始终是网页设计追求的目标，通过 JavaScript 可以实现多个网页特效。文字的打字效果是 JavaScript 脚本程序，将预先设置好的文字逐一在页面上显示出来。具体步骤如下所示。

01 分析需求。如果要在网页中实现打字效果，需要创建预先设置好的文字，作为输出信息。该实例完成效果如图 18-1 所示。

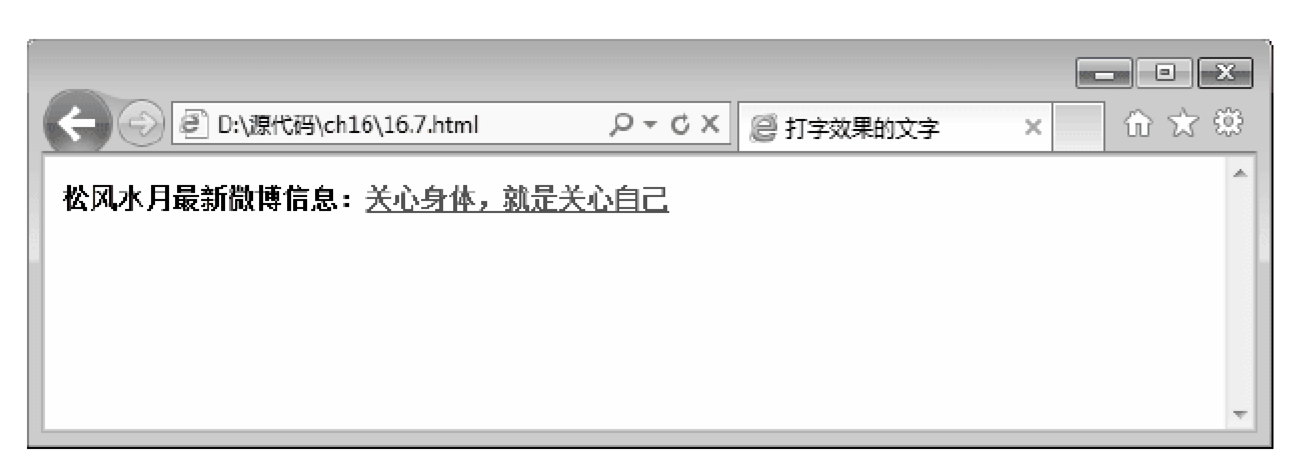


图 18-1　打字效果

02 创建 HTML 页面，设置页面基本样式，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>打字效果的文字</title>
<style type="text/css">
body{font-size:14px;font-weight:bold;}
</style>
</head>
<body>
松风水月最新微博信息：<a id="HotNews" href="" target="_blank"></a>
```

```
</body>
</html>
```

上面代码中，在<head>标记中间设置 body 页面的基本样式，例如字体大小为 14 像素，字形加粗。并在 body 页面创建了一个超级链接。

在 IE 9.0 中浏览效果如图 18-2 所示，可以看到页面中只显示了一个提示信息。



图 18-2　页面基本样式

添加 JavaScript 代码，实现打字特效：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
var NewsTime = 2000;   //每条微博的停留时间
var TextTime = 50;       //微博文字出现等待时间，越小越快
var newsi = 0;
var txti = 0;
var txttimer;
var newstimer;
var newstitle = new Array();   //微博标题
var newshref = new Array();      //微博链接
newstitle[0] = "健康是身体的本钱";
newshref[0] = "#";
newstitle[1] = "关心身体，就是关心自己";
newshref[1] = "#";
newstitle[2] = "去西藏旅游了";
newshref[2] = "#";
newstitle[3] = "大雨倾盆，很大呀";
newshref[3] = "#";
function shownew()
{
   var endstr = "_"
   hwnewstr = newstitle[newsi];
   newslink = newshref[newsi];
   if(txti==(hwnewstr.length-1)){endstr="";}
   if(txti>=hwnewstr.length){
      clearInterval(txttimer);
      clearInterval(newstimer);
```

```
        newsi++;
        if(newsi>=newstitle.length){
          newsi = 0
        }
        newstimer = setInterval("shownew()",NewsTime);
        txti = 0;
        return;
      }
      clearInterval(txttimer);
      document.getElementById("HotNews").href=newslink;
      document.getElementById("HotNews").innerHTML = hwnewstr.substring(0,txti+1)+endstr;
      txti++;
      txttimer = setInterval("shownew()",TextTime);
    }
    shownew();
    //-->
    </SCRIPT>
```

因为上面代码是一个整体，这里就不分开介绍了。在 JavaScript 代码中，主要调用 shownew()函数完成打字效果。在 JavaScript 代码开始部分，定义了多个变量，其中数组对象 newstitle 用于存放文本标题。下面创建了 shownew()函数，并在函数中通过变量和条件获取要显示的文字，通过“setInterval("shownew()",NewsTime)”语句输出文字内容。代码最后使用 shownew()语句循环执行该函数中的输出信息。

在 IE 9.0 中浏览效果如图 18-3 所示，可以看到页面中每隔一定时间，会在提示信息后，逐个打出单个文字，字体颜色为蓝色。



图 18-3　实现打字效果

18.2　综合实例 2——文字升降特效

有的网页为了加大广告宣传力度，往往在网页上设置一个自动升降的文字，用于吸引人的注意力。当单击这个升降文字，会自动跳转到宣传页面。本实例将使用 JavaScript 和 CSS 实现文字升降效果。具体步骤如下所示。

01 分析需求。如果需要实现文字升降，需要指定文字内容和文字升降范围，即为文字

在 HTML 页面指定一个层，用于升降文字。实例完成后，实际效果如图 18-4 所示。



图 18-4　文字升降

02 创建 HTML，构建升降 DIV 层，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>升降的文字效果</title>
</head>
<body>
<div id="napis" style="position: absolute;top: -50;color: #000000;font-family:宋体;font-size:9pt;border:1px #ddeecc solid">
<a href="" style="font-size:12px;text-decoration:none;">
水月大酒店，欢迎天下来宾！
</a></div>
<script language="JavaScript">
<!--
setTimeout('start()',20);
//-->
</script>
</body>
</html>
```

上面代码创建了一个 DIV 层，用于存放升降的文字，层的 ID 名称是 napis，并在层的 style 属性中定义了层显示样式。例如字体大小、带有边框、字形等。在 DIV 层中创建了超级链接，并设定了超级链接的样式。其中的 script 代码用于定时调用 start 函数。在 IE 9.0 中浏览效果如图 18-5 所示，可以看到页面空白，无文字显示。



图 18-5　空白页面

03 添加 JavaScript 代码，实现文字升降，代码如下：

```
<script language="JavaScript">
<!--
done = 0;
step = 4
function anim(yp,yk)
{
if(document.layers) document.layers["napis"].top=yp;
else document.all["napis"].style.top=yp;
if(yp>yk) step = -4
if(yp<60) step = 4
setTimeout('anim('+(yp+step)+','+yk+')', 35);
}function start()
{
if(done) return
done = 1;
if(navigator.appName=="Netscape") {
var nap=document.getElementById("napis");
nap.left=innerWidth/2 - 145;
anim(60,innerHeight - 60)
}
else {
napis.style.left=11;
anim(60,document.body.offsetHeight - 60)
}}//-->
</script>
```

上面代码创建了 anim()和 start()函数，其中 anim()函数用于设定每次升降的数值，start()函数用于设定每次开始的升降的坐标。在 IE 9.0 中浏览效果如图 18-6 所示，可以看到页面中超级链接自动上下移动。



图 18-6　上下移动

18.3　综合实例 3——跑马灯效果

网页中有一种特效称为跑马灯，即文字从左向右自动输出，和晚上写字楼的广告霓虹灯非常相似。在网页中，如果 CSS 样式设计非常完美，就会产生更加靓丽的网页效果。具体步骤

如下：

01 分析需求。完成跑马灯效果，需要使用 JavaScript 语言设置文字内容、移动速度和相应输入框，使用 CSS 设置显示文字样式。输入框用来显示水平移动文字。实例完成后，实际效果如图 18-7 所示。



图 18-7　马灯效果

02 创建 HTML，实现输入表单，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>跑马灯</title>
  </head>
<body onLoad="LenScroll()">
<center>
<form name="nextForm">
<input type=text name="lenText">
</form>
</center>
</body>
```

上面代码非常简单，先创建表单，并在表单中存放入文本域，用于显示移动文字。在 IE 9.0 中浏览效果如图 18-8 所示，可以看到页面中只是存在一个文本域，没有其他显示信息。

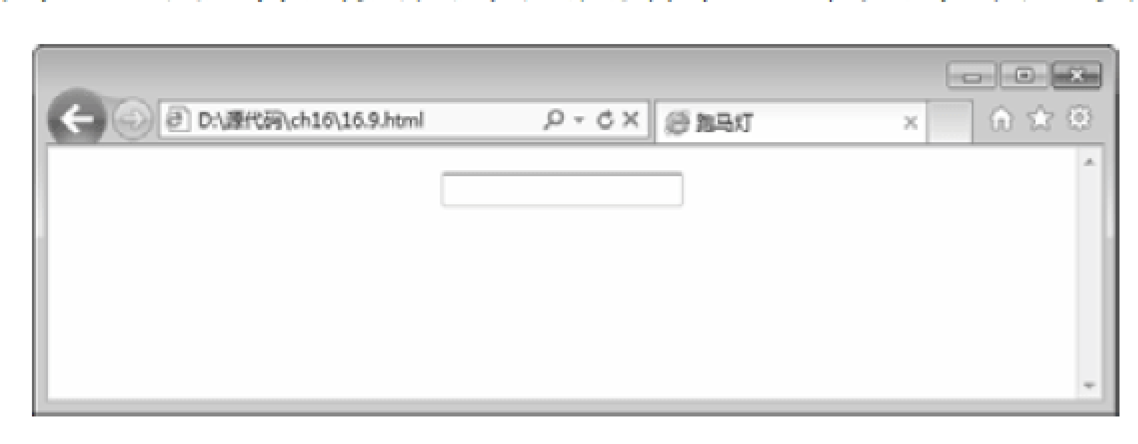


图 18-8　实现基本表单

03 添加 JavaScript 代码，实现文字移动，代码如下：

```
<script language="JavaScript">
var msg="品味中原文化，寄情黄河风景";      //移动文字
var interval = 400;                      //移动速度
var seq=0;
```

```
function LenScroll() {
    document.nextForm.lenText.value = msg.substring(seq, msg.length) + "      " + msg;
    seq++;
    if ( seq >msg.length )
        seq = 0;
    window.setTimeout("LenScroll();", interval);
}
</script>
```

上面代码中，创建了一个变量 msg，用于定义移动的文字内容，变量 interval 用于定义文字移动速度，LenScroll()函数用于在表单输入框中显示移动信息。

在 IE 9.0 中浏览效果如图 18-9 所示，可以看到输入框中显示了移动信息，并且从右向左移动。



图 18-9　实现移动效果

04 添加 CSS 代码，修饰输入框和页面，代码如下：

```
<style type="text/css">
<!--
body{
    background-color:#FFFFFF;   /* 页面背景色 */
}
input{
    background:transparent;       /* 输入框背景透明 */
    border:none;              /* 无边框 */
    color:#ffb400;
    font-size:45px;
    font-weight:bold;
    font-family:黑体;
}--></style>
```

上面代码设置了页面背景颜色为白色，在 input 标记选择器中，定义了边框背景为透明，无边框，字体颜色为黄色，大小为 45 像素，加粗、黑体显示。在 IE 9.0 中浏览效果如图 18-10 所示，可以看到页面中相比较原来页面字体变大，颜色为黄色，没有输入框显示。



图 18-10 最终效果

18.4 综合实例 4——闪烁图片

图片闪烁是常用的一种特效，用 JavaScript 实现起来非常简单，这时需要注意时间间隔这个参数，数值越大闪烁越不连续，数值越小闪烁越历害，可以随意更改这个值，直到取得满意效果。具体步骤如下所示。

01 分析需求。将图片放在一个 DIV 层上，设定图片为可见的，然后使用 JavaScript 程序代码设置 DIV 层的显示和隐藏，这样就达到了图片的闪烁效果。实例完成后，效果如图 18-11 所示。



图 18-11 闪烁效果

02 创建 HTML 页面，构建 DIV 层，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>闪烁图片</title>
</head>
<body ONLOAD="soccerOnload()" topmargin="0">
<DIV ID="soccer" STYLE="position:absolute; left:150; top:0">
<a href="">
<IMG SRC="feng.jpg" border="0"></a>
</DIV>
</body>
</html>
```

上面代码中，创建了一个层，其 ID 名称为 soccer，样式为绝对定位，坐标位置为(150,0)。

然后在层中创建一个图片，不带边框。在 IE 9.0 中浏览效果如图 18-12 所示，可以看到显示一个图片，不具有闪烁效果。



图 18-12　图片

03 添加 JavaScript 代码，实现图片闪烁，代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
var msecs = 500; //改变时间得到不同的闪烁间隔；
var counter = 0;
function soccerOnload() {
setTimeout("blink()", msecs);
}
function blink() {
soccer.style.visibility =
(soccer.style.visibility == "hidden") ? "visible" : "hidden";
counter +=1;
setTimeout("blink()", msecs);
}
</SCRIPT>
```

在 JavaScript 代码中，创建变量 msecs 用于定义闪烁时间间隔，创建变量 counter 用于计数。在函数 soccerOnload()中设定每隔指定时间图片闪烁一次，函数 blink()用于设定图片显示方式，即层是隐藏还是可见。

在 IE 9.0 中浏览效果如图 18-13 所示，可以看到显示一个图片，在指定时间内闪烁。



图 18-13　最终效果

18.5 综合实例 5——左右移动的图片

在广告栏中，经常会有从右到左移动或者从左到右移动的图片，一张或者多张，不但增加了页面效果，也有提升广告效应的作用。本实例将使用 JavaScript 和 CSS 创建一个左右移动的图片。具体步骤如下所示。

01 分析需求。实现左右移动的图片，需要再页面上定义一张图片，然后利用 JavaScript 程序代码，获取图片对象，并使其在一定范围内，即水平方向上自由移动。实例完成，效果如图 18-14。



图 18-14 图片移动

02 创建 HTML 页面，导入图片，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>左右移动图片</title>
</head>
<body>
<img src="feng.jpg" name="picture"
style="position: absolute; top: 70px; left: 30px;" BORDER="0" WIDTH="140" HEIGHT="40">
<script LANGUAGE="JavaScript"><!--
setTimeout("moveLR('picture',300,1)",10);
//--></script>
</body>
</html>
```

上面代码中，定义了一个图片，为绝对定位，左边位置是（70,30）无边框，宽度为 140 像素，高度为 40 像素。script 标记中，使用 setTimeout 方法，定时移动图片。

在 IE 9.0 中浏览效果如图 18-15 所示，可以看到网页上显示一个图片。

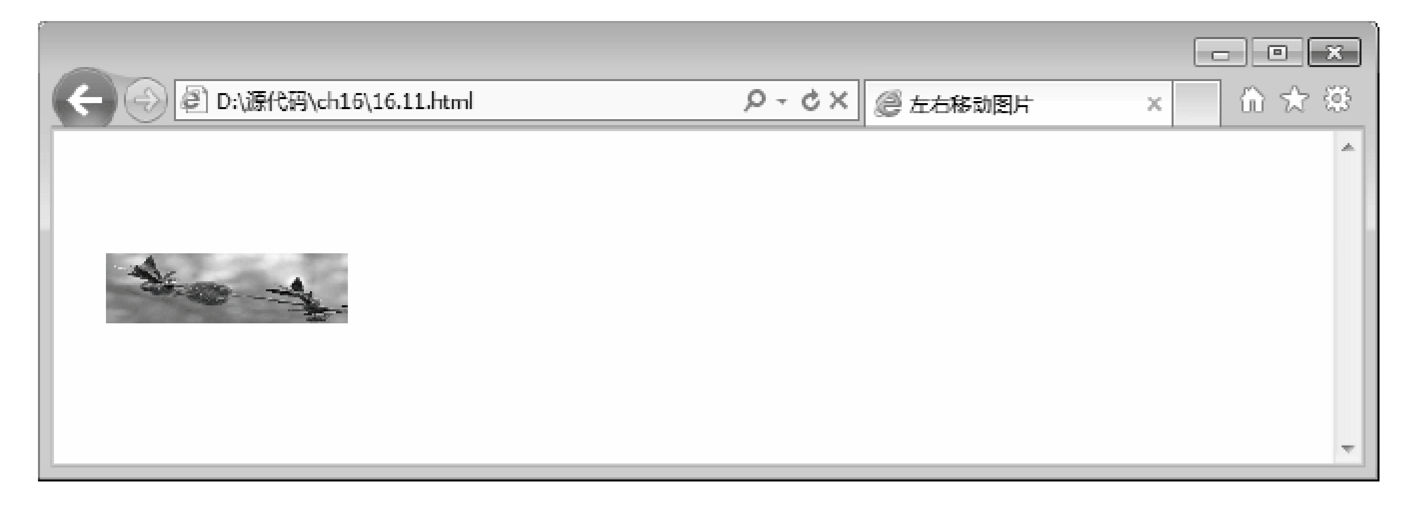


图 18-15　图片显示

03 加入 JavaScript 代码，实现图片左右移动，代码如下：

```
<script LANGUAGE="JavaScript"><!--
step = 0;
obj = new Image();
function anim(xp,xk,smer) //smer = direction
{
obj.style.left = x;
x += step*smer;
if (x>=(xk+xp)/2) {
if (smer == 1) step--;
else step++;
}
else {
if (smer == 1) step++;
else step--;
}
if (x >= xk) {
x = xk;
smer = -1;
}
if (x<= xp) {
x = xp;
smer = 1;
}
// if (smer >2) smer = 3;
setTimeout('anim('+xp+','+xk+','+smer+')', 50);
}
function moveLR(objID,movingarea_width,c)
{
if (navigator.appName=="Netscape") window_width = window.innerWidth;
else window_width = document.body.offsetWidth;
obj = document.images[objID];
image_width = obj.width;
x1 = obj.style.left;
```

```
x = Number(x1.substring(0,x1.length-2)); // 30px ->30
if (c == 0) {
if (movingarea_width == 0) {
right_margin = window_width - image_width;
anim(x,right_margin,1);
}
else {
right_margin = x + movingarea_width - image_width;
if (movingarea_width< x + image_width) window.alert("No space for moving!");
else anim(x,right_margin,1);
}
}
else {
if (movingarea_width == 0) right_margin = window_width - image_width;
else {
x = Math.round((window_width-movingarea_width)/2);
right_margin = Math.round((window_width+movingarea_width)/2)-image_width;
}
anim(x,right_margin,1);
}
}
//--></script>
```

上面代码和文字水平方向移动原理基本相同，只是对象不同，这里不再介绍。

在 IE 9.0 中浏览效果如图 18-16 所示，可以看到网页上显示一个图片，并在水平方向上自由移动。



图 18-16　最终效果

18.6　综合实例 6——向上滚动菜单

网页中包含信息比较多的时候，需要设计出一些导航菜单，来实现页面导航。如果使用 JavaScript 代码将菜单做成动态效果，此时菜单会更加吸引人。本实例将结合前面学习的内容，创建一个向上滚动的菜单。具体步骤如下：

01 分析需求。实现菜单自动从下到上滚动，需要把握两个元素，一个是使用 JavaScript 实现要滚动的菜单，即导航栏，另一个是使用 JavaScript 控制菜单移动方向。实例完成，效果如图 18-17 所示。



图 18-17　菜单滚动

02 构建 HTML 页面，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>向上滚动的菜单</title>
</head>
<body bgcolor="#FFFFFF" text="#000000">
</body></html>
```

上面代码比较简单，只实现了一个空白页面，页面背景色为白色，前景色为黑色。在 IE 9.0 中浏览效果如图 18-18 所示，可以看到显示了一个空白页面。

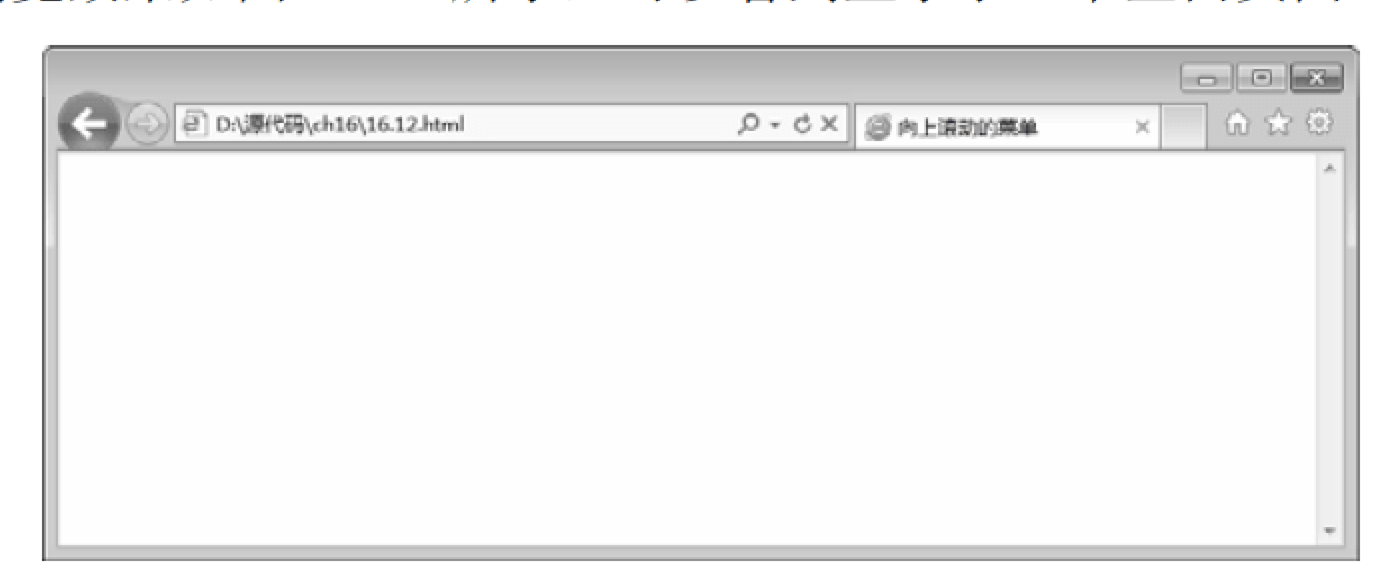


图 18-18　空白 HTML 页面

03 加入 JavaScript 代码，实现菜单滚动，代码如下：

```
<script language=JavaScript>
<!--
  var index = 9
  link = new Array(8);
  link[0] ='time1.htm'
  link[1] ='time2.htm'
  link[2] ='time3.htm'
  link[3] ='time1.htm'
```

```
        link[4] ='time2.htm'
        link[5] ='time3.htm'
        link[6] ='time1.htm'
        link[7] ='time2.htm'
        link[8] ='time3.htm'
        text = new Array(8);
        text[0] ='首页'
        text[1] ='产品天地'
        text[2] ='关于我们'
        text[3] ='资讯动态'
        text[4] ='服务支持'
        text[5] ='会员中心'
        text[6] ='网上商城'
        text[7] ='官方微博'
        text[8] ='企业文化'
        document.write ("<marquee scrollamount='1' scrolldelay='100' direction= 'up' width='150' height='150'>");
        for (i=0;i<index;i++)
        {
          document.write (" <img src='dian3.gif' width='12' height='12'><a href="+link[i]+" target='_blank'>");
          document.write (text[i] + "</A><br>");
        }
        document.write ("</marquee>")
    // --></script>
```

上面代码创建了两个数组对象 link 和 text，用来存放菜单链接对象和菜单内容，在 JavaScript 代码中，使用<marquee>定义页面在垂直方向上下移动。

在 IE 9.0 中浏览效果如图 18-19 所示，可以看到面左侧有一个菜单，自下向上自由移动。



图 18-19 最终效果

18.7 综合实例 7——跟随鼠标移动的图片

在众多网站中，特别是游戏网站或小型商业网站，都喜欢用鼠标图片跟随的特效，一方面可以在鼠标指针旁边加上网站说明的相关信息或者欢迎信息；另一方面也吸引人的注意力，使

其更加关注此类网站。本实例实现图片跟随鼠标移动的特效，具体步骤如下所示。

01 分析需求。需要通过 JavaScript 获取鼠标指针的位置，并且动态的调整图片的位置。图片需要通过 position 的绝对定位，很容易得到调整。采用 CSS 的绝对定位是 JavaScript 调整页面元素常用的方法，效果如图 18-20 所示。



图 18-20　图片移动

02 创建基本 HTML 页面，代码如下：

```
<!DOCTYPE html>
<html >
<head>
<title>随鼠标移动的图片</title>
</head>
<body>
</body>
</html>
```

03 添加 JavaScript 代码，实现图片随鼠标移动，代码如下：

```
<script type="text/JavaScript">
function badAD(html){
    var ad=document.body.appendChild(document.createElement('div'));
    ad.style.cssText="border:1px solid #000;background:#FFF;position:absolute;padding:4px 4px 4px 4px;font: 12px/1.5 verdana;";
    ad.innerHTML=html||'This is bad idea!';
    var c=ad.appendChild(document.createElement('span'));
    c.innerHTML="×";
    c.style.cssText="position:absolute;right:4px;top:2px;cursor:pointer";
    c.onclick=function (){
        document.onmousemove=null;
        this.parentNode.style.left='-99999px'
    };
    document.onmousemove=function (e){
        e=e||window.event;
        var x=e.clientX,y=e.clientY;
```

```
                setTimeout(function() {
                        if(ad.hover)return;
                        ad.style.left=x+5+'px';
                        ad.style.top=y+5+'px';
                },120)
        }
        ad.onmouseover=function (){
                this.hover=true
        };
        ad.onmouseout=function (){
                this.hover=false
        }
}
badAD('<img src="18.png">')
</script>
```

上面代码中，使用 appendChild()方法为当前页面创建了一个 DIV 对象，并为 DIV 层设置了相应样式。下面的 e.clientX 和 e.clientY 语句确定鼠标位置，并动态调整图片位置，从而实现图片移动效果。在 IE 9.0 中浏览效果如图 18-21 所示，可以看到鼠标在页面移动时，图片跟着移动。



图 18-21　最终效果

18.8　综合实例 8——树形菜单

作为一个首页，其特点之一是需要导航的页面很多，有时为了效果不得不将所有需要导航的部分都放到一个导航菜单中。树形导航菜单是网页设计中最常用的菜单之一。本实例将创建一个树形菜单，具体步骤如下所示。

01 分析需求。实现一个树形菜单，需要三个方面配合，一个是<ul>无序列表，用于显示的菜单，一个是 CSS 样式，修饰树形菜单样式，一个是 JavaScript 程序，实现单击时展开菜单选项。实例完成后，效果如图 18-22 所示。

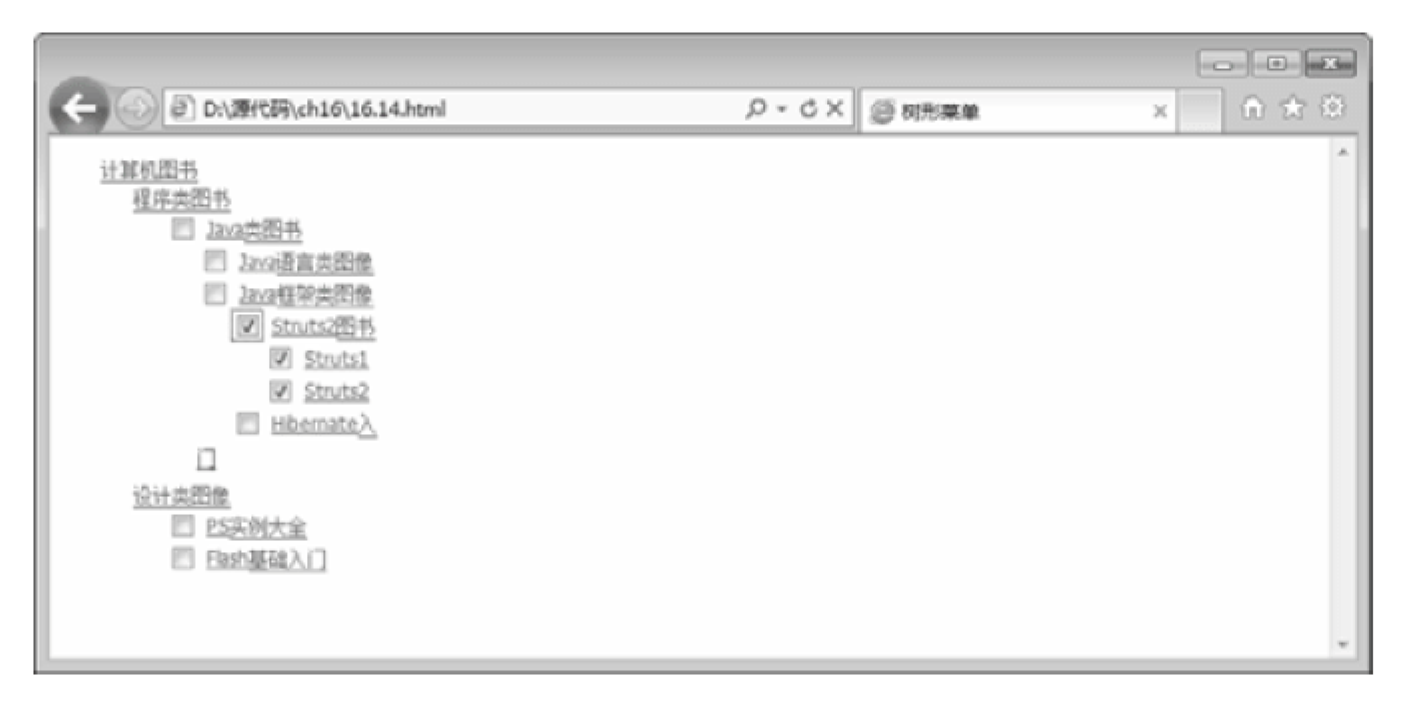


图 18-22　树形菜单

02 创建 HTML 页面，实现菜单列表，代码如下：

```
<!DOCTYPE html>
<html >
<head>
<title>树形菜单</title>
</head>
<body>
<ul id="menu_zzjs_net">
<li>
 <label><a href="JavaScript:;">计算机图书</a></label>
 <ul class="two">
  <li>
   <label><a href="JavaScript:;">程序类图书</a></label>
   <ul class="two">
    <li>
     <label><input type="checkbox" value="123456"><a href="JavaScript:;">Java 类图书</a></label>
     <ul class="two">
      <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">Java 语言类图像
</a></label></li>
      <li>
       <label><input type="checkbox" value="123456"><a href="JavaScript:;">Java 框架类图像
</a></label>
       <ul class="two">
        <li>
         <label><input type="checkbox" value="123456"><a href="JavaScript:;">Struts2 图书</a></label>
         <ul class="two">
          <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">Struts1</a></label>
</li>
          <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">Struts2</a></label>
</li>
         </ul>
```

```
            </li>
            <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">Hibernate 入 门
</a></label></li>
           </ul>
          </li>
         </ul>
        </li>
       </ul>
      </li>
      <li>
       <label><a href="JavaScript:;">设计类图像</a></label>
       <ul class="two">
        <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">PS 实例大全</a></label>
</li>
        <li><label><input type="checkbox" value="123456"><a href="JavaScript:;">Flash 基 础 入 门 </a>
</label></li>
       </ul>
      </li>
     </ul>
    </li>
    </ul>
    </body>
    </html>
```

在 IE 9.0 中浏览效果如图 18-23 所示，可以看到无序列表在页面上显示，并且显示全部元素，字体颜色为蓝色。

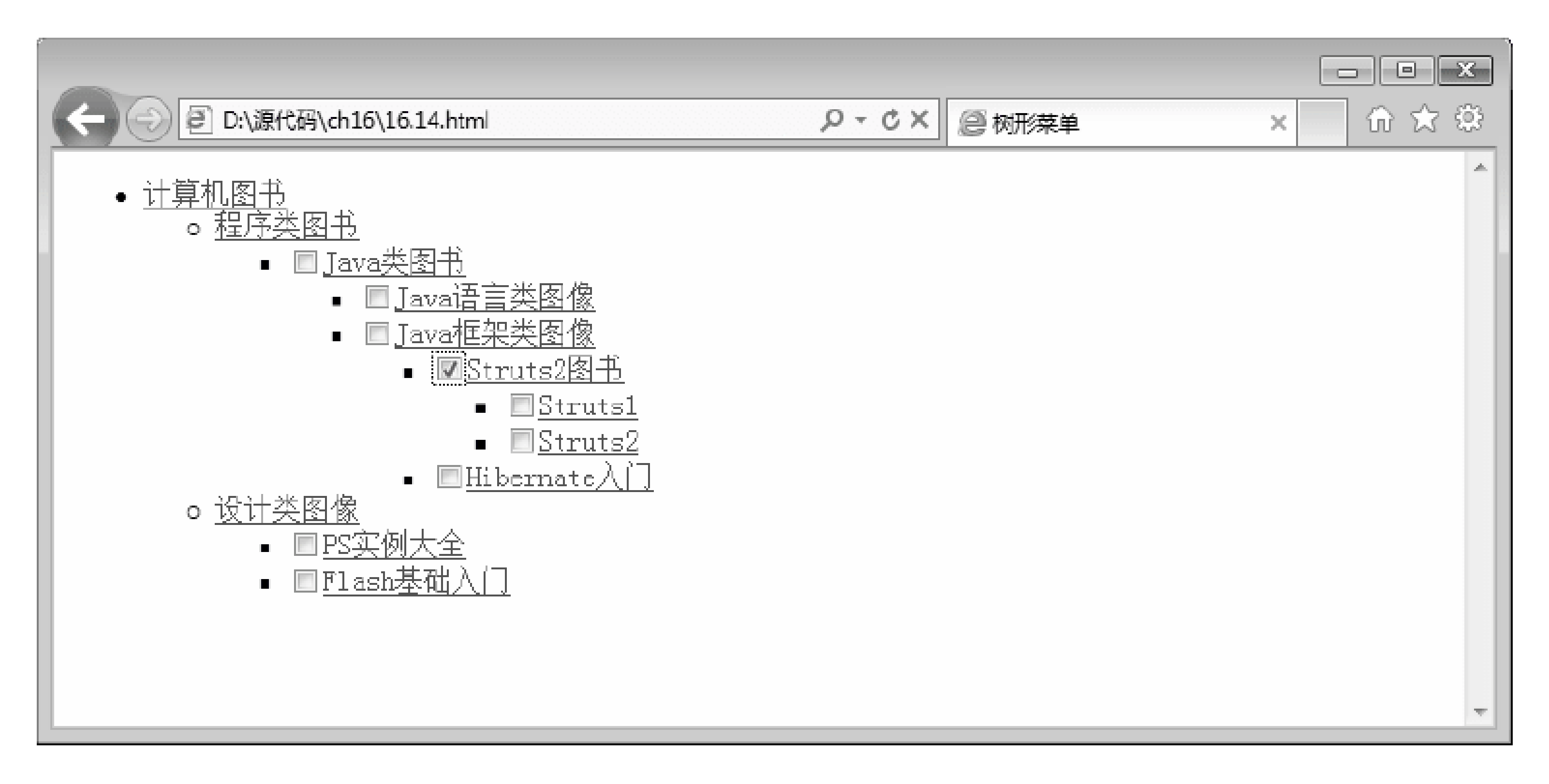


图 18-23　无序列表

03 添加 JavaScript 代码，实现单击展开，代码如下：

```
    <script type="text/JavaScript" >
     function addEvent(el,name,fn){//绑定事件
      if(el.addEventListener) return el.addEventListener(name,fn,false);
```

```
        return el.attachEvent('on'+name,fn);
    }
    function nextnode(node){//寻找下一个兄弟并剔除空的文本节点
        if(!node)return ;
        if(node.nodeType == 1)
            return node;
        if(node.nextSibling)
            return nextnode(node.nextSibling);
    }
    function prevnode(node){//寻找上一个兄弟并剔除空的文本节点
        if(!node)return ;
        if(node.nodeType == 1)
            return node;
        if(node.previousSibling)
            return prevnode(node.previousSibling);
    }
    function parcheck(self,checked){//递归寻找父亲元素，并找到 input 元素进行操作
        var par =   prevnode(self.parentNode.parentNode.parentNode.previousSibling),parspar;
        if(par&&par.getElementsByTagName('input')[0]){
            par.getElementsByTagName('input')[0].checked = checked;
            parcheck(par.getElementsByTagName('input')[0],sibcheck(par.getElementsByTagName('input')[0]));
        }
    }
    function sibcheck(self){//判断兄弟节点是否已经全部选中
        var sbi = self.parentNode.parentNode.parentNode.childNodes,n=0;
        for(var i=0;i<sbi.length;i++){
            if(sbi[i].nodeType != 1)//由于孩子结点中包括空的文本节点，所以这里累计长度的时候也要算上去
                n++;
            else if(sbi[i].getElementsByTagName('input')[0].checked)
                n++;
        }
        return n==sbi.length?true:false;
    }
    addEvent(document.getElementById('menu_zzjs_net'),'click',function(e){//绑定 input 单击事件，使用
menu_zzjs_net 根元素代理
        e = e||window.event;
        var target = e.target||e.srcElement;
        var tp = nextnode(target.parentNode.nextSibling);
        switch(target.nodeName){
            case 'A'://单击 A 标签展开和收缩树形目录，并改变其样式会选中 checkbox
                if(tp&&tp.nodeName == 'UL'){
                    if(tp.style.display != 'block' ){
```

```
            tp.style.display = 'block';
            prevnode(target.parentNode.previousSibling).className = 'ren'
          }else{
            tp.style.display = 'none';
            prevnode(target.parentNode.previousSibling).className = 'add'
          }
        }
        break;
      case 'SPAN'://单击图标只展开或者收缩
        var ap = nextnode(nextnode(target.nextSibling).nextSibling);
        if(ap.style.display != 'block' ){
          ap.style.display = 'block';
          target.className = 'ren'
        }else{
          ap.style.display = 'none';
          target.className = 'add'
        }
        break;
      case 'INPUT'://单击 checkbox，父亲元素选中，则孩子节点中的 checkbox 也同时选中，孩子结点取消父元素随之取消
        if(target.checked){
          if(tp){
            var checkbox = tp.getElementsByTagName('input');
            for(var i=0;i<checkbox.length;i++)
              checkbox[i].checked = true;
          }
        }else{
          if(tp){
            var checkbox = tp.getElementsByTagName('input');
            for(var i=0;i<checkbox.length;i++)
              checkbox[i].checked = false;
          }
        }
        parcheck(target,sibcheck(target));//当孩子结点取消选中的时候调用该方法递归其父节点的 checkbox 逐一取消选中
        break;
    }
  });
  window.onload = function(){//页面加载时给有孩子结点的元素动态添加图标
    var labels = document.getElementById('menu  zzjs  net').getElementsByTagName('label');
    for(var i=0;i<labels.length;i++){
      var span = document.createElement('span');
```

```
        span.style.cssText ='display:inline-block;height:18px;vertical-align:middle;width:16px;cursor:pointer;';
        span.innerHTML = ' '
        span.className = 'add';
        if(nextnode(labels[i].nextSibling)&&nextnode(labels[i].nextSibling).nodeName == 'UL')
          labels[i].parentNode.insertBefore(span,labels[i]);
        else
          labels[i].className = 'rem'
      }
    }
</script>
```

在 IE 9.0 中浏览效果如图 18-24 所示，可以看到无序列表在页面上显示，使用鼠标单击可以展开或关闭相应的选项，但其样式非常难看。

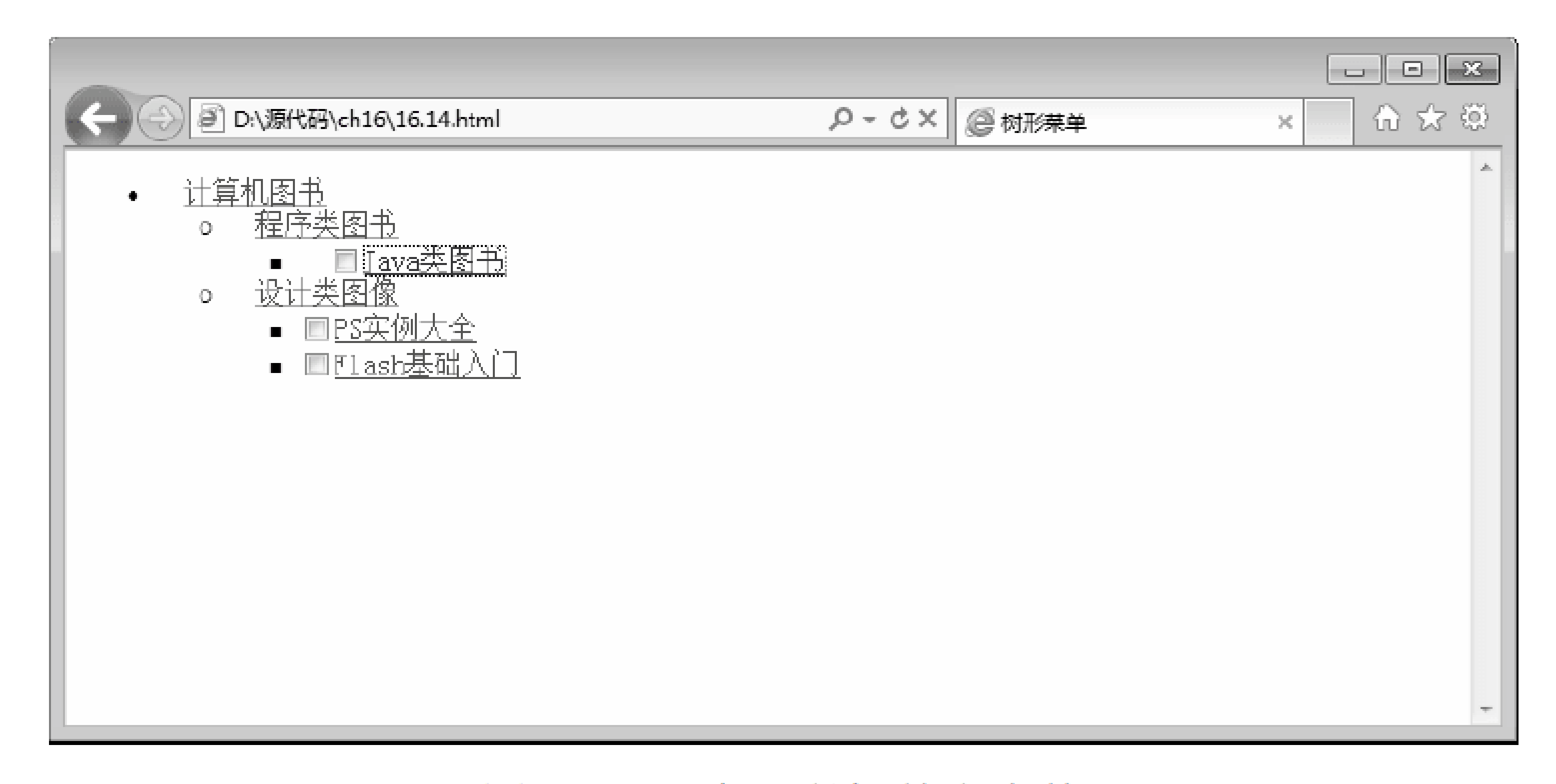


图 18-24　实现鼠标单击事件

04 添加 CSS 代码，修饰列表选项，代码如下：

```
<style type="text/css">
body{margin:0;padding:0;font:12px/1.5 Tahoma,Helvetica,Arial,sans-serif;}
ul,li,{margin:0;padding:0;}
ul{list-style:none;}
#menu_zzjs_net{margin:10px;width:200px;overflow:hidden;}
#menu_zzjs_net li{line-height:25px;}
#menu_zzjs_net .rem{padding-left:16px;}
#menu_zzjs_net .add{background:url() -4px -31px no-repeat;}
#menu_zzjs_net .ren{background:url() -4px -7px no-repeat;}
#menu_zzjs_net li a{color:#666666;padding-left:5px;outline:none;blr:expression(this.onFocus=this.blur());}
#menu_zzjs_net li input{vertical-align:middle;margin-left:5px;}
#menu_zzjs_net .two{padding-left:20px;display:none;}
</style>
```

在 IE 9.0 中浏览效果如图 18-25 所示，可以看到样式变得非常漂亮。



图 18-25 最终效果

18.9 综合实例 9——时钟特效

在 HTML 5 技术中，新增了一个容器画布 canvas，用来在页面上绘制一些图形，利用这个新的特性，可以在网页中创建一个类似于钟表的特效。本实例结合第 2 章知识，创建了一个时钟特效。具体步骤如下所示。

01 分析需求。在画布上绘制时钟，需要绘制几个必要的图形，表盘、时针、分针、秒针和中心圆这几个图形。这样将上面几个图形组合起来，便构成了一个时针界面，然后使用 JavaScript 代码，根据时间移动秒针、分针和时针。实例完成后，效果如图 18-26 所示。



图 18-26 时钟特效

02 创建 HTML 页面，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>canvas 时钟</title>
</head><body>
<canvas id="canvas" width="200" height="200" style="border:1px solid #000;">您的浏览器不支持 Canvas。</canvas></body></html>
```

上面代码创建了一个画布，其宽度为 200 像素，高度为 200 像素，带有边框，颜色为黑色，

样式为直线型。在 Firefox 5.0 中浏览效果如图 18-27 所示，可以看到显示了一个带有黑色边框的画布，画布中没有任何信息。

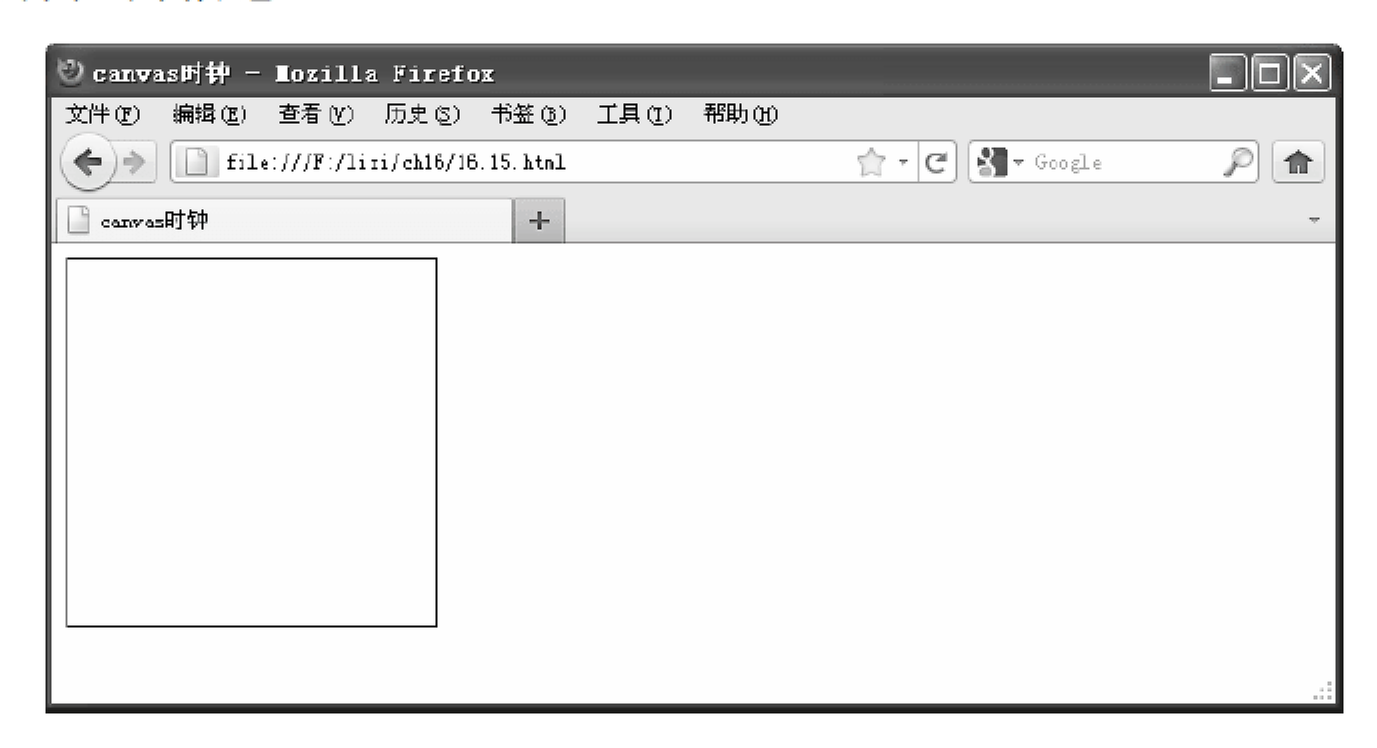


图 18-27　定义画布

03 添加 JavaScript，绘制不同图形，代码如下：

```
<script type="text/JavaScript" language="JavaScript" charset="utf-8">
 var canvas = document.getElementById('canvas');
 var ctx = canvas.getContext('2d');
 if(ctx){
  var timerId;
  var frameRate = 60;
  function canvObject(){
   this.x = 0;
   this.y = 0;
   this.rotation = 0;
   this.borderWidth = 2;
   this.borderColor = '#000000';
   this.fill = false;
   this.fillColor = '#ff0000';
   this.update = function(){
    if(!this.ctx)throw new Error('你没有指定 ctx 对象。');
    var ctx = this.ctx
    ctx.save();
    ctx.lineWidth = this.borderWidth;
    ctx.strokeStyle = this.borderColor;
    ctx.fillStyle = this.fillColor;
    ctx.translate(this.x, this.y);
    if(this.rotation)ctx.rotate(this.rotation * Math.PI/180);
    if(this.draw)this.draw(ctx);
    if(this.fill)ctx.fill();
    ctx.stroke();
    ctx.restore();
```

```
        }   };............
      timerId = setInterval(function(){
        // 清除画布
        ctx.clearRect(0,0,200,200);
        // 填充背景色
        ctx.fillStyle = 'orange';
        ctx.fillRect(0,0,200,200);
        // 表盘
        circle.update();
        // 刻度
        for(var i=0;cache=ls[i++];)cache.update();
        // 时针
        hour.rotation = (new Date()).getHours() * 30;
        hour.update();
        // 分针
        minute.rotation = (new Date()).getMinutes() * 6;
        minute.update();
        // 秒针
        seconds.rotation = (new Date()).getSeconds() * 6;
        seconds.update();
        // 中心圆
        center.update();
      },(1000/frameRate)|0);
    }else{
      alert('您的浏览器不支持 Canvas 无法预览.\n 跟我一起说："很遗憾!"');
    }
  </script>
```

由于篇幅比较长，这里只显示了部分代码。其详细代码可以在光盘中查询。上面代码首先绘制了不同类型的图形，例如时针、秒针和分针等。然后再将其组合在一起，并根据时间定义时针等指向。在 Firefox 5.0 中浏览效果如图 18-28 所示，可以看到页面中出现了一个时钟，其秒针在不停地移动。

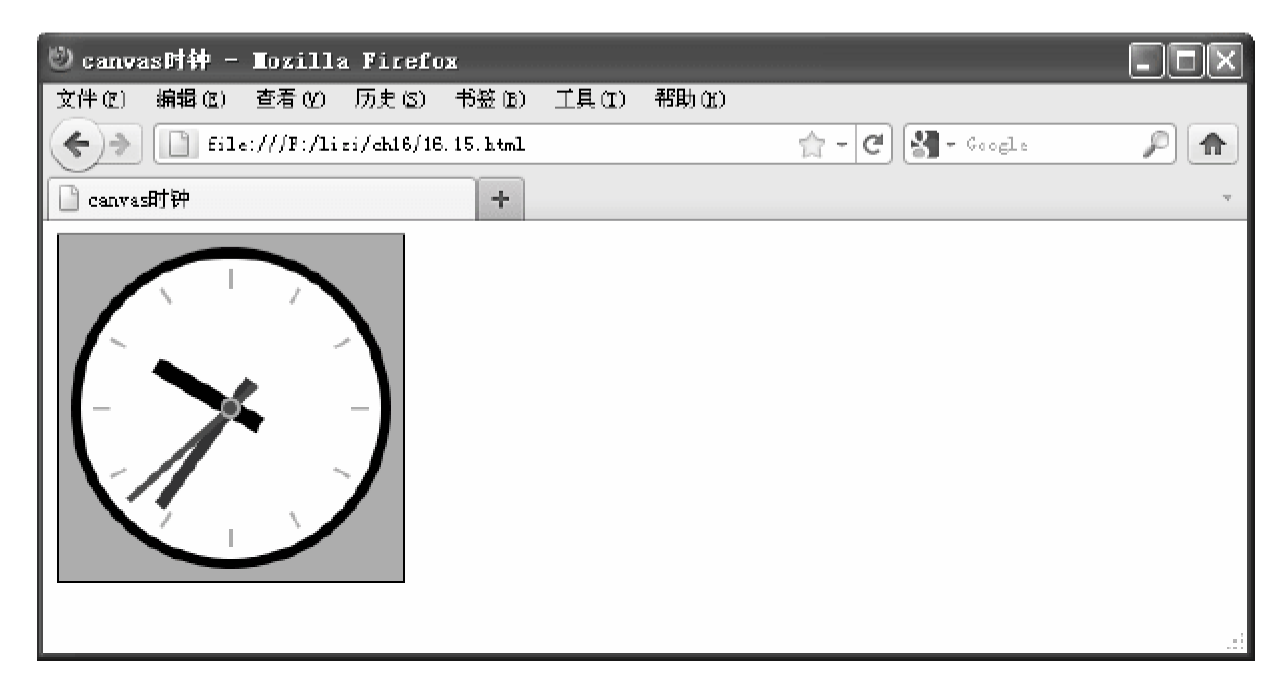


图 18-28　最终特效

18.10　综合实例 10——颜色选择器

在页面中定义背景色和字体颜色，是比较常见的一种操作，在选取颜色时比较麻烦，不知道那种颜色适合，并且不知道颜色值是什么。此时可以利用颜色选择器来定义颜色并获取颜色值。本实例将创建一个颜色选择器，可以自由获取颜色值。具体步骤如下：

01 分析需求。本实例原理非常简单，就是将几个常用的的颜色值进行组合，再合并，就是所要选择的颜色值，是利用 JavaScript 代码完成的。实例完成后，实际效果如图 18-29 所示。

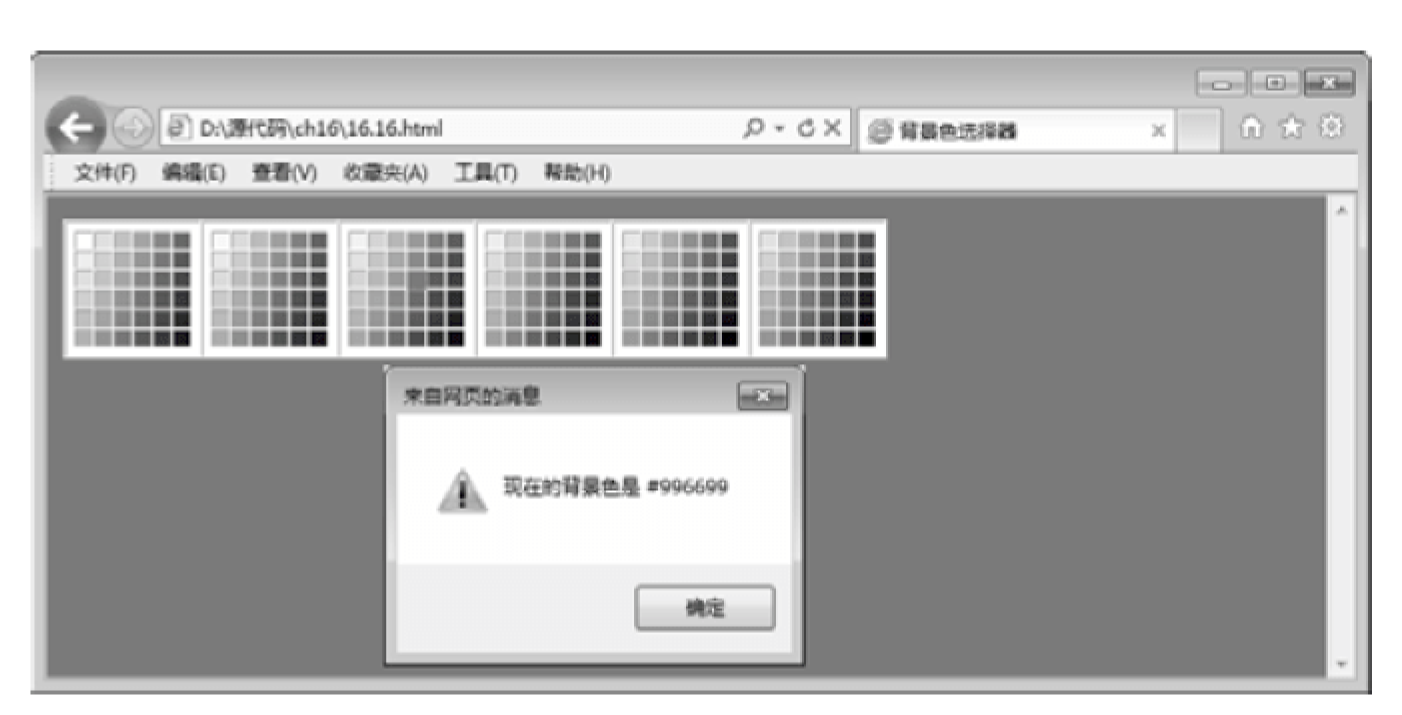


图 18-29　设定页面背景色

02 创建基本 HTML 页面，代码如下：

```
<!DOCTYPE html>
<html>
<head><title>背景色选择器</title>
</head>
<body bgcolor="#FFFFFF">
</body></html>
```

03 添加 JavaScript 代码，实现颜色选择，代码如下：

```
<script language="JavaScript">
<!--
var hex = new Array(6)
hex[0] = "FF"
hex[1] = "CC"
hex[2] = "99"
hex[3] = "66"
hex[4] = "33"
hex[5] = "00"
function display(triplet)
{
  document.bgColor = '#' + triplet
```

```
    alert('现在的背景色是 #'+triplet)
  }
  function drawCell(red, green, blue)
  {
    document.write('<TD BGCOLOR="#' + red + green + blue + '">')
    document.write('<A HREF="JavaScript:display(\'' + (red + green + blue) + '\')">')
    document.write('<IMG SRC="place.gif" BORDER=0 HEIGHT=12 WIDTH=12>')
    document.write('</A>')
    document.write('</TD>')
  }
  function drawRow(red, blue)
  {
    document.write('<TR>')
    for (var i = 0; i< 6; ++i)
    {
      drawCell(red, hex[i], blue)
    }   document.write('</TR>')
  }function drawTable(blue)
  {
    document.write('<TABLE CELLPADDING=0 CELLSPACING=0 BORDER=0>')
    for (var i = 0; i< 6; ++i)
    {
      drawRow(hex[i], blue)
    }
    document.write('</TABLE>')
  }
  function drawCube()
  {
    document.write('<TABLE CELLPADDING=5 CELLSPACING=0 BORDER=1><TR>')
    for (var i = 0; i< 6; ++i)
    {
      document.write('<TD BGCOLOR="#FFFFFF">')
      drawTable(hex[i])
      document.write('</TD>')
    }   document.write('</TR></TABLE>')
  }drawCube()
  // --></script>
```

上面代码中，创建了一个数组对象 hex 用来存放不同的颜色值。下面几个函数分别将数组中颜色组合在一起，并在页面显示，display 函数完成定义背景颜色和显示颜色值。

在 IE 9.0 中浏览效果如图 18-30 所示，可以看到页面显示多个表格，每个单元格代表一种颜色。



图 18-30 最终效果

18.11 问题解答

1．JavaScript 的数组中常用方法有哪些？

- jion()：将 array 中的所有 element 以 string 的形式连在一起：

```
var a = [1,2,3];
s = a.join();          // s == "1,2,3"
s = a.join(": ");   // s == "1: 2: 3"
```

- reverse()：将 Array 的 element 顺数颠倒：

```
var a = [1,2,3];
a.reverse();
s = a.join(); // s == "3,2,1"
```

- sort()：排序，默认按字母顺序排序 case sensitive，可以自定义排序方式.：

```
var a = [111,4,33,2];
a.sort();   // a == [111,2,33,4]
a.sort(function(a,b) { // a == [2,4,33,111]
        return a-b;            // Returns< 0, 0, or >0
  });
```

- concat()：连接多个 Array：

```
var a = [1,2,3];
a.concat(4,5);                    // return [1,2,3,4,5]
a.concat([4,5]);                 // return [1,2,3,4,5]
a.concat([4,5], [6,7])     // return [1,2,3,4,5,6,7]
a.concat(4,[5,[6,7]]);    // return [1,2,3,4,5,6,7]
```

2．JavaScript 中注释有哪些？

在程序中添加注释来描述代码的功能，通过注释还可以使一些代码无效，以实现逐行检查

程序的目的，可及时发现并解决问题。JavaScript 主要提供了三种注释的方法：

- 单行注释：在需要注释的代码前添加字符“// “，//后面的部分会被注释。
- 多行注释：在代码前添加“/*”，之后添加“*/”，之间的部分会被注释。
- HTML 注释：使用传统的 HTML 注释，“<!—”和“-->”之间的部分会被注释，注释内容可以是一行或多行。

第 19 章　综合实战——企业门户网站

作为小型软件企业的网站，一般规模不是太大，通常包含 3~5 个栏目，例如“产品”、“客户”和“联系我们”栏目等，并且有的栏目甚至只包含一个页面，例如“联系我们”栏目。此类网站通常都是为展示公司形象，说明公司业务范围和产品特色的，这样的网站只要一个首页加上若干内容即可。

19.1　构思布局

本实例是模拟一个小型软件公司的网站，其公司主要承接电信方面的各种软件项目。网站上包括“首页”、“产品”、“客户”和“联系方式”等栏目。本实例中采用红色和白色配合，红色部分显示导航菜单，白色显示文本信息。在 IE 9.0 中浏览其效果如图 19-1 所示。



图 19-1　计算机网站首页

19.1.1　设计分析

作为一个软件公司网站首页，其页面需要简单、明了，给人以清晰的感觉。页头部分主要放置导航菜单和公司 Logo 信息等，其 Logo 可以是一张图片或者文本信息等；页面主体分为两个部分，页面主体左侧是公司介绍，可以在首页上概括性描述；页面主体右侧是“最新消息”、“产品展示”和“客户”等，其中“产品展示”和“客户”的链接信息，以列表形式介绍重要信息，也可以通过页面顶部导航菜单进入相应页面介绍。

对于网站的其他子页面，篇幅可以比较短，其重点是介绍软件公司业务、联系方式、产品

信息等，页面需要与首页风格相同。

19.1.2 排版架构

从上面效果图可以看出，页面结构并不是太复杂，采用的是上中下结构，页面主体部分又嵌套了一个左右版式结构。其效果如图 19-2 所示。

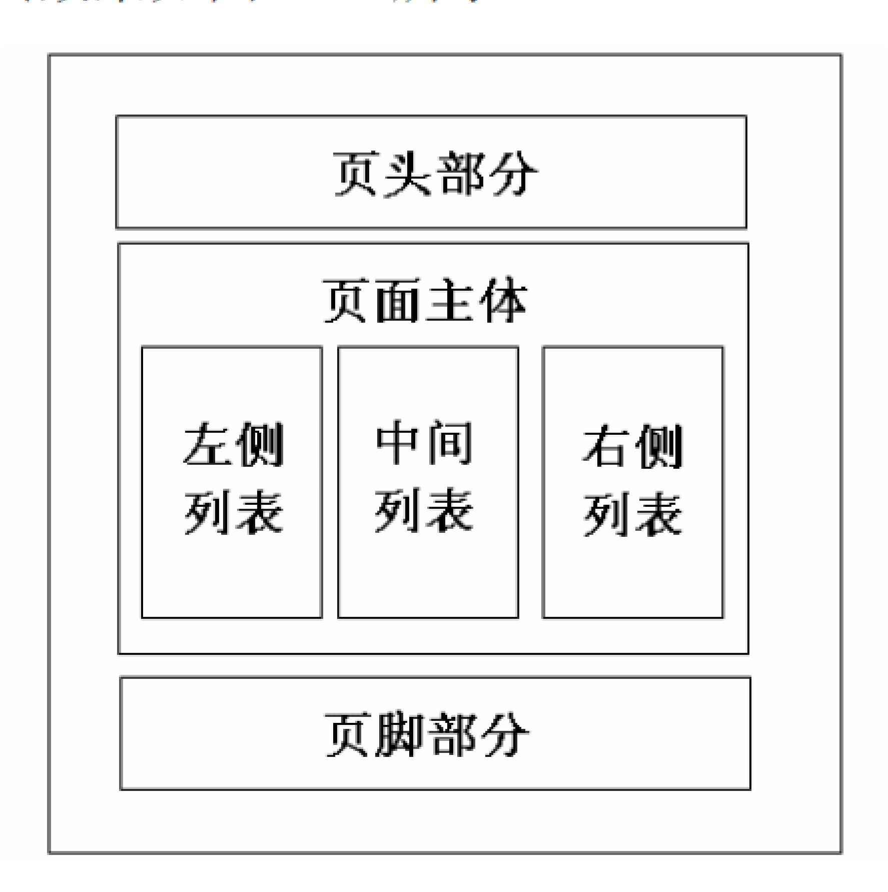


图 19-2 页面总体框架

在 HTML 页面中，通常使用 DIV 层对应上面不同的区域，可以是一个 DIV 层对应一个区域，也可以是多个 DIV 层对应同一个区域。本实例的 DIV 代码如下：

```
<div id="container">/*页面布局容器*/
<div id="top">
</div><!--end top-->
<div id="header">
</div><!--end header-->
<div id=me>/*导航菜单*/
</div>
<div id="content">
<div id="text">/*页面主体左侧内容*/
</div><!--end text-->
<div id="column">/*页面主体右侧内容*/
</div><!--end column-->
</div><!--end content-->
<div id="footer">/*页脚部分*/
</div><!--end footer-->
</div><!--end container-->
```

上面代码中，id 名称为 container 的层是整个页面的布局容器，top 层、header 层和 me 层共同组成了页头部分，top 层用于显示页面 logo，header 层用于显示页头文本信息，me 层用于显示页头导航菜单信息。页面主体是 content 层，其包含了两个层，text 层和 column 层，text 层是页面主体左侧的内容，显示公司介绍信息；column 层同页面主体右侧的内容，显示公司

常用的导航链接。Footer 层是页脚部分，用于显示版权信息和地址信息。

在 CSS 文件中，对应 container 层和 content 层的 CSS 代码如下：

```
#container
{
 margin: 0pt auto;
width: 770px;
 position: relative; }
#content {
 background: transparent url('images/content.gif') repeat-y;
 clear: both;
 margin-top: 5px;
width: 770px;
}
```

上面代码中，#container 选择器定义了整个布局容器的宽度、外边距和定位方式。#content 选择器定义了背景图片、宽度和顶部边距。

19.2 模块分割

当页面整体架构完成后，就可以动手制作不同的模块区域。其制作流程采用自上而下，从左到右的顺序。完成后，再对页面样式进行整体调整。

19.2.1 Logo 与导航菜单

一般情况下，Logo 信息和导航菜单都放在页面顶部，作为页头部分。其中 Logo 信息作为公司标志，通常放在页面的左上角或右上角；导航菜单放在页头部分和页面主体二者之间，用于链接其他的页面。在 IE 9.0 中浏览效果如图 19-3 所示。



图 19-3 页面 Logo 和导航菜单

在 HTML 文件中，用于实现页头部分的 HTML 代码如下：

```
<div id="top">
</div><!--end top-->
<div id="header">
<h1>计算机  网站</h1>
</div><!--end header-->
<div id=me>
```

```
<ul id="menu">
<li><a href="#" class="actual">首页</a></li>
<li><a href="#" >产品</a></li>
<li><a href="#">客户</a></li>
<li><a href="#">联系方式</a></li>
</ul>
</div>
```

上面代码中，top 层用于显示页面 Logo，header 层用于显示页头的文本信息，例如公司名称；me 层用于显示页头导航菜单。在 me 层中，有一个无序列表，用于制作导航菜单，每个选项都由超级链接组成。

在 CSS 样式文件中，对应上面标记的 CSS 代码如下：

```
#top {
background: transparent url('images/top.jpg') no-repeat;
height: 50px;
}
#top p {
 margin: 0pt;
padding: 0pt;
}
#header {
 background: transparent url('images/header.jpg') no-repeat;
 height: 150px;
 margin-top: 5px;
}
#menu {
 position: absolute;
top: 180px;
left: 15px;
}
#header h1 {
 margin: 5px 0pt 0pt 50px;
padding: 0pt;
font-size: 1.7em;
 }
#header h2 {
margin: 10px 0pt 0pt 90px;
 padding: 0pt;
 font-size: 1.2em;
color: rgb(223, 139, 139);
}
```

```
ul#menu {
margin: 0pt;
}
#menu li {
list-style-type: none;
float: left;
text-align: center;
 width: 104px;
margin-right: 3px;
 font-size: 1.05em;
}
#menu a {
 background: transparent url('images/menu.gif') no-repeat;
overflow: hidden;
 display: block;
height: 28px;
padding-top: 3px;
 text-decoration: none;
 twidth: 100%;
font-size: 1em;
font-family: Verdana,"Geneva CE",lucida,sans-serif;
color: rgb(255, 255, 255);
}
#menu li >a, #menu li >strong {
 width: auto;
 }
#menu a.actual {
background: transparent url('images/menu-actual.gif') no-repeat;
color: rgb(149, 32, 32);
}
#menu a:hover {
 color: rgb(149, 32, 32);
 }
```

上面代码中，#top 选择器定义了背景图片和层高；#header 定义了背景图片、高度和顶部外边距；#menu 层定义了层定位方式和坐标位置。其他选择分别定义了上面三个层中元素的显示样式，例如段落显示样式、标题显示样式和超级链接样式等。

19.2.2 左侧文本介绍

在页面主体中，其左侧版式主要介绍公司相关信息。左侧文本采用的是左浮动、固定宽度的版式设计，重点在于调节宽度使不同浏览器之间能够效果一致，并且颜色上要配合 Logo 和左侧的导航菜单，使整个网站和谐、大气。在 IE 9.0 中浏览效果如图 19-4 所示。

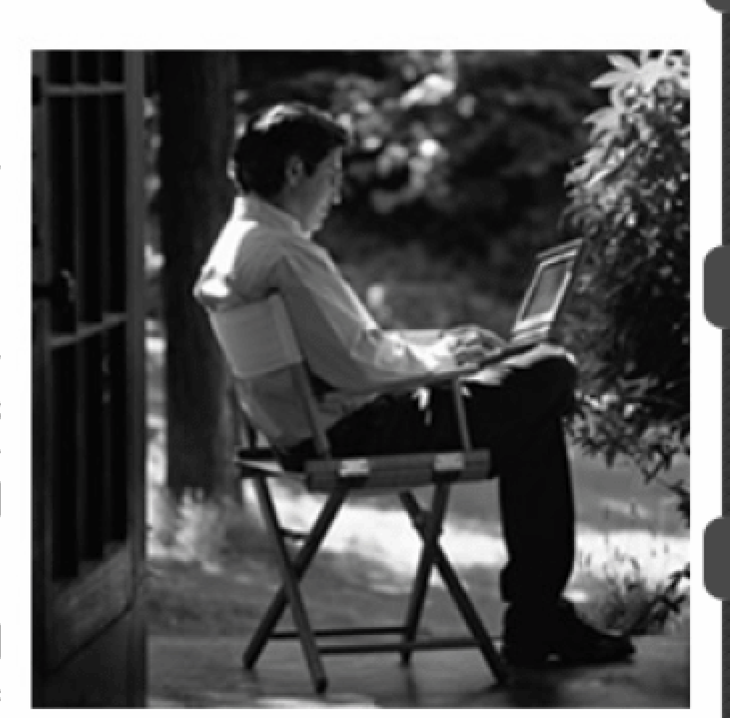


图 19-4　页面左侧文本介绍

在 HTML 文件中，创建页面左侧内容介绍的代码如下：

```
<div id="content">
<div id="text">
<h3 class="headlines"><a href="#" title="testing">欢迎来到我们的网站</a></h3>
<p><img src="images/fotos.jpg" alt="fotos" align="right" />
远大公司成立于 1998 年，注册资本 1700 万元。是国家认定的高新技术企业、软件企业，是专业的电信系统仿软件和应用服务供应商。</p><p>
公司坚持走自立创新、稳步发展的道路，以创立品牌为自己的基本策略，以产品自身的品质，先进的技术和良好的服务取信于用户。2002 年至今公司先后有多个软件产品获得了河南省信息产业厅颁发的《软件产品登记证书》和国家版权局颁发的《软件著作权登记证书》。同时远大的进步和发展，也得到了政府部门的大力支持和关注，获得国家科技部和省、市政府部门技术创新基金无偿资助百余万元。并正式获得中国质量体系认证中心颁发的 ISO9001:2008 质量管理体系认证证书。
</p>
<p> </p>
</div><!--end text-->
</div>
```

上面代码中，content 层是页面主体，text 层是页面主体中左侧的部分。text 层包含了标题和段落信息，段落中包含一张图片。

在 CSS 文件中，对于上面 HTML 标记的 CSS 代码，如下所示：

```
#text {
background: rgb(255, 255, 255) url('text-top.gif') no-repeat;
 width: 518px;
color: rgb(0, 0, 0);
float: left;
}
```

```
#text h1, #text h2, #text h3, #text h4 {
  color: rgb(140, 9, 9);
}
#text h3.headlines a {
  color: rgb(140, 9, 9);
}
```

上面代码中，#text 层定义了背景图片、背景颜色、字体颜色和页面左浮动。下面选择器定义了标题显示样式，例如字体颜色等。“#text h3.headlines a”选择器定义了标题 3、类 headlines 和超级链接显示样式。

19.2.3　右侧导航链接

在页面主体右侧版式中，其文本信息不是太多，但非常重要。是首页用于链接其他页面的导航链接，例如客户详细信息、最新消息等。同样，右侧版式需要设置为固定宽度并且向左浮动的版式。在 IE 9.0 中浏览页面效果如图 19-5 所示。

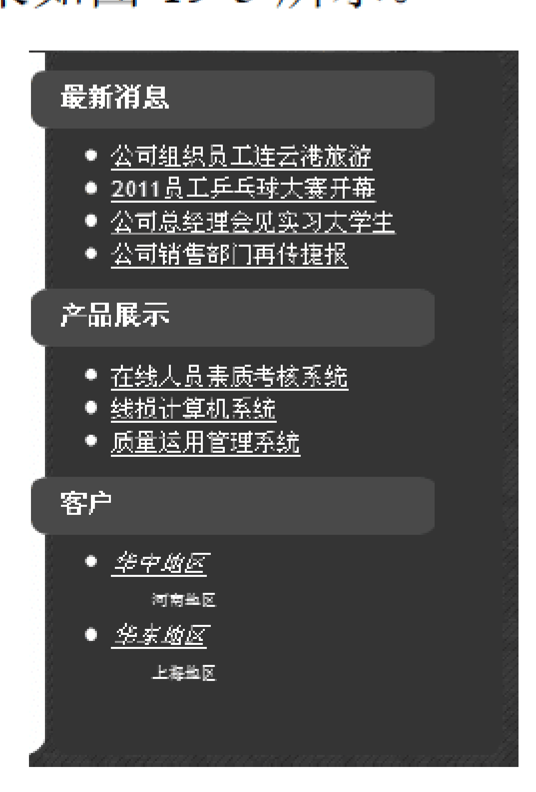


图 19-5　页面右侧链接

从上面效果图可以，需要包含几个无序列表和标题，其中列表选项为超级链接。HTML 文件中用于创建页面主体右侧版式的代码，如下所示：

```
<div id="column">
<h3><span>最新消息</span></h3>
<ul class="category_list"><li><a href="#">公司组织员工连云港旅游</a></li>
<li><a href="#">2011 员工乒乓球大赛开幕</a></li>
<li><a href="#">公司总经理会见实习大学生</a></li>
<li><a href="#">公司销售部门再传捷报</a></li></ul>
<h3><span>产品展示</span></h3>
<ul class="recent_articles"><li><a href="#">在线人员素质考核系统</a></li>
<li><a href="#">线损计算机系统</a></li>
<li><a href="#">质量运用管理系统</a></li></ul>
<h3><span>客户</span></h3>
<ul class="wet_recent_comments"><li><a href="#"><cite>华中地区</cite></a><p>河南地区</p></li>
```

```
<li><a href="#"><cite>华东地区</cite></a><p>上海地区</p></li></ul>
</div><!--end column-->
<div id="content-bottom"> </div>
```

在上面的代码中，创建了两个层，分别为 column 层和 content-bottom 层。其中 column 层用于显示页面主体中右侧链接，并包含了三个标题和三个超级链接。content-bottom 层用于消除 float 层的浮动效果。

在 CSS 文件中，用于修饰上面 HTML 标记的 CSS 代码，如下所示：

```
#column {
  background: rgb(142, 14, 14) url('images/column.gif') no-repeat;
float: right;
  width: 247px; }
#column p { font-size: 0.7em; }
#column ul { font-size: 0.8em; }
#column h3 {
background: transparent url('images/h3-column.gif') no-repeat;
position: relative;
left: -18px;
  height: 26px;
width: 215px;
margin-top: 10px;
padding-top: 6px;
padding-left: 6px;
font-size: 0.9em;
z-index: 1;
font-family: Verdana,"Geneva CE",lucida,sans-serif;
}
#column h3 span { margin-left: 10px; }
#column span.name {
  text-align: right;
  color: rgb(223, 58, 0);
margin-right: 5px;
}
#column a { color: rgb(255, 255, 255); }
#column a:hover { color: rgb(80, 210, 122); }
p.comments {
text-align: right;
font-size: 0.8em;
  font-weight: bold;
  padding-right: 10px;
}
```

```
#content-bottom {
background: transparent url('images/content-bottom.gif') no-repeat scroll left bottom;
clear: both;
 display: block;
width: 770px;
height: 13px;
 font-size: 0pt;
}
```

上面代码中，#column 选择器定义背景图片、背景颜色、页面右浮动和宽度。#content-bottom 选择器定义背景图片、宽度、高度、字体大小和以块显示，并且使用 clear 消除前面层使用 float 的影响。其他选择器主要定义 column 层中其他元素的显示样式，例如无序列表样式、列表选项样式和超级链接样式等。

19.2.4 版权信息

版权信息一般放置到页面底部，用于介绍页面的作者、地址信息等，是页脚的一部分。页脚部分和其他网页部分一样，需要保持简单、清晰的风格。在 IE 9.0 中浏览效果如图 19-6 所示。

图 19-6　页脚部分

从图 19-6 可以看出，此页脚部分非常简单，只包含了一个作者信息的超级链接，因此设置起来比较方便，其代码如下：

```
<div id="footer">
<p id="ivorius"><a href="#">网页设计者：李四工作室</a></p>
</div><!--end footer-->
```

上面代码中，层 footer 包含了一个段落信息，其中段落的 id 是 ivorius。

在 CSS 文件中，用于修饰上面 HTML 标记的样式代码，如下所示：

```
#footer {
 background: transparent url('images/footer.png') no-repeat scroll left bottom;
margin-top: 5px;
padding-top: 2px;
height: 33px;
}
#footer p { text-align: center; }
#footer a { color: rgb(255, 255, 255); }
#footer a:hover { color: rgb(223, 58, 0); }
p#ivorius {
float: right;
```

```
    margin-right: 13px;
     font-size: 0.75em;
    }
    p#ivorius a { color: rgb(80, 210, 122); }
```

上面代码中，#footer 选择器定义了页脚背景图片、内外边距的顶部距离和高度。其他选择器定义了页脚部分文本信息的对齐方式、超级链接样式等。

19.3 整体调整

前面的各个小节中，完成了首页中不同部分的制作，其整个页面基本上都已经成形。在制作完成后，需要根据页面实际效果作一些细节上的调整，从而能更加完善页面整体效果。例如各块之间的 padding 和 margin 值是否与页面整体协调，各个子块之间是否协调统一等。页面效果调整前，在 IE 9.0 中浏览效果如图 19-7 所示。



图 19-7 页面调整前效果

从图 19-7 可以发现页面段落没有缩进，页面右侧列表选项之间距离太小等。这时可以利用 CSS 属性调整，其代码如下所示：

```
p { margin: 0.4em 0.5em; font-size: 0.85em;text-indent:2em; }
a { color: rgb(25, 126, 241); text-decoration: underline; }
a:hover { color: rgb(223, 58, 0); text-decoration: none; }
a img { border: medium none ; }
ul, ol { margin: 0.5em 2.5em; }
h2 { margin: 0.6em 0pt 0.4em 0.4em; }
h3, h4, h5 { margin: 1em 0pt 0.4em 0.4em; }
```

```
* { margin: 0pt; padding: 0pt; }
body { background: rgb(61, 62, 63) url('images/body.gif') repeat; color: white; font-size: 1em; font-family: "Trebuchet MS",Tahoma,"Geneva CE",lucida; }
```

上面代码中，全局选择器“*”设置了内外边距距离，body 标记选择器设置了背景颜色、图片、字体大小，字体颜色和字形等。其他选择器分别设置了段落、超级链接、标题和列表等样式信息。

19.4 问题解答

1．Firefox 和 IE 浏览器，如何处理负边界问题？

在 IE 中，对于超出父元素的部分会被父元素覆盖，而在 Firefox 中，对于超出父元素的部分会覆盖父元素，但前提是父元素有边框或内边距，不然负边距会显示在父元素上，使得父元素拥有负边距。在进行网页设计时，针对上面的情况可以对元素进行相对定位。

2．在定义子元素的上边距时，如果超出元素高度，怎么处理？

在 IE 浏览器中，子元素上边距显示正常，而在 Firefox 浏览器中，子元素上边距显示在父元素上方。其解决办法是在父元素中增加 overflow:hidden 语句或给父元素增加边框或内边距。

第 20 章　综合实战——HTML 5 游戏

HTML 5 作为下一代 Web 开发标准，成为主流指日可待。它对音频视频、事件处理、绘图等的支持都让人们非常期待。本章主要通过一个 HTML 5 游戏，让读者轻松掌握 HTML 5、CSS 和 JavaScript 的新特性，并能灵活应用到 Web 开发中。

20.1　游戏概述

本实例将制作一个星际争霸的游戏，需要实现的效果如下：

玩家按 Enter 键开始游戏，进入游戏主界面后，按键盘上的左右键控制游戏的移动方向，按 A 键开始射击。默认情况下，玩家的生命值为 100 分，如果玩家没有射击到敌机，则分为两种情况：敌机和玩家的战机相撞，则减少玩家生命值 10 分；敌机和玩家的战机没有相撞，则玩家的生命值不减少。

游戏开始页面的效果如图 20-1 所示。



图 20-1　游戏开始界面

按 Enter 键，进入游戏运行页面，按左右键控制方向，按 A 键射击敌机，效果如图 20-2

所示。

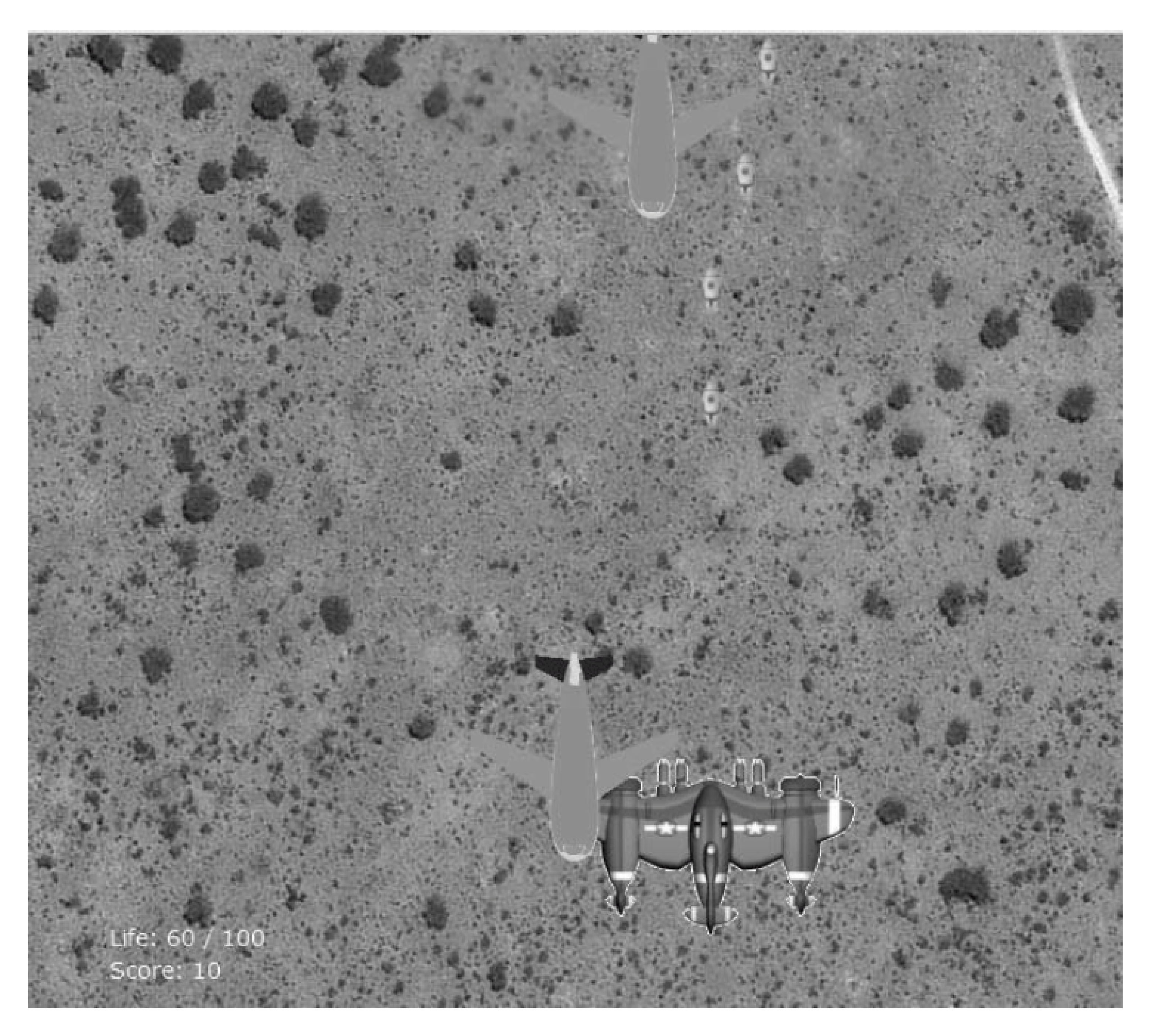


图 20-2　游戏主界面

20.2　游戏需求分析

创建星际争霸游戏主要的文件夹列表如图 20-3 所示。

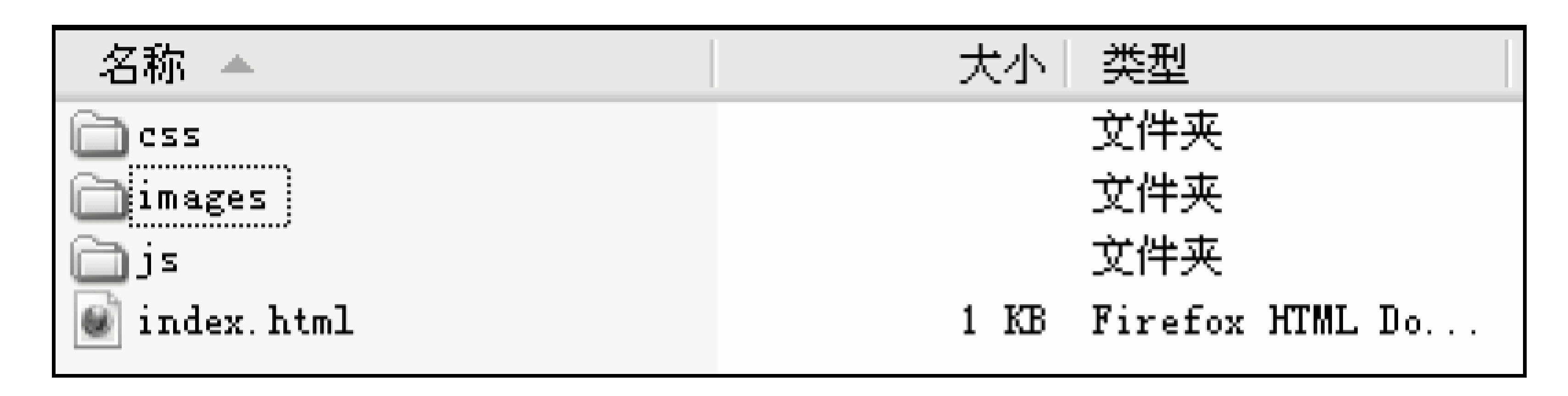


图 20-3　游戏的主要文件夹列表

其中 css 文件夹下主要包括游戏网页的样式表，文件名称为 main.css。images 文件夹下主要包括游戏的飞机和环境背景图等。js 文件夹下主要包括 jquery.js 和 script.js 两个文件。其中 jquery.js 主要定义了 JavaScript 脚本的框架，script.js 文件主要是为了实现游戏功能。Index.html 是游戏启动的主页面。

Images 文件夹下的图片效果如图 20-4 所示。

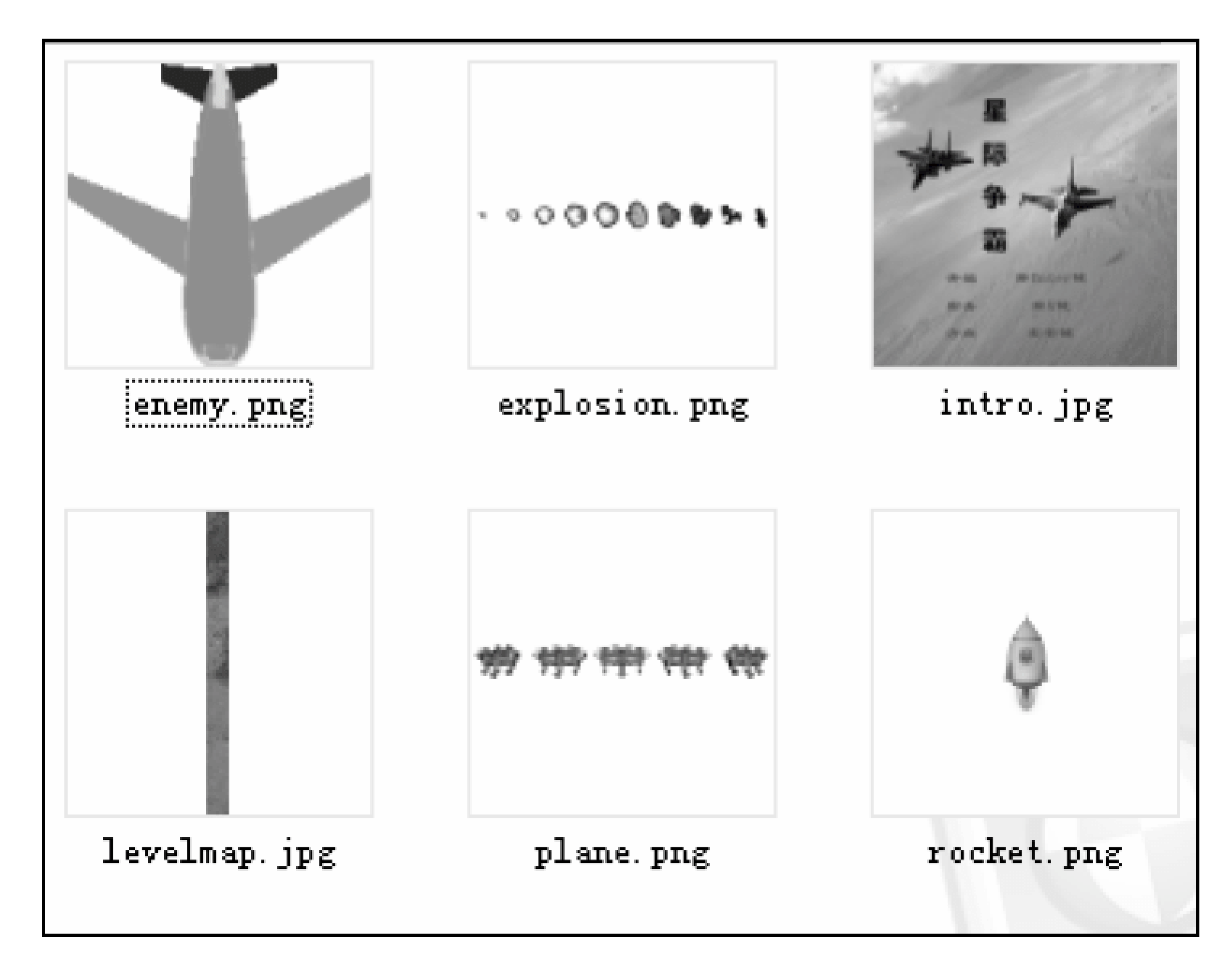


图 20-4　游戏需要的主要图片

20.3　HTML 5、CSS 和 JavaScript 搭配实现

下面查看游戏实现所需要的 HTML 5、CSS 和 JavaScript 的技术。

20.3.1　基本的 HTML 5 结构和标记

用户首先需要创建 Index.html，实现基本的结构和标记。具体的代码如下：

```
<!DOCTYPE html>
<html lang="en" >
    <head>
        <meta charset="GB2312" />
        <title>星际争霸游戏</title>
        <!-- 添加样式表 -->
        <link href="css/main.css" rel="stylesheet" type="text/css" />
        <!-- 添加 scripts 脚本代码-->
        <!--[if lt IE 9]>
          <script src="http://html 5shiv.googlecode.com/svn/trunk/html 5.js"></script>
        <![endif]-->
        <script src="js/jquery.js"></script>
        <script src="js/script.js"></script>
    </head>
    <body>
        <header>
            <h2>星际争霸</h2>
        </header>
```

```
        <div class="container">
            <canvas id="scene" width="700" height="700" tabindex="1"></canvas>
        </div>
    </body>
</html>
```

其中<!--[if lt IE 9]>表示如果浏览器版本小于 IE9。则加载<script src="http://HTML5shiv.googlecode.com/svn/trunk/HTML 5.js"></script>JavaScript 脚本代码。<script src="js/jquery.js"></script>和<script src="js/script.js"></script>代码表示加载 JavaScript 脚本。<canvas id="scene" width="700" height="700" tabindex="1"></canvas>表示绘图的宽高为 700×700 像素，tabindex="1"表示键盘移动顺序。

运行效果如图 20-5 所示。



图 20-5　游戏主界面

HTML 5 最重要的新特征主要包括 canvas。这个新技术为开发人员提供了一个新途径，可以采用一种完全自由的方式绘图、包含图像以及放置文本。与以前的 HTML 版本相比，这是一个比较显著的改进。在以前的版本中，虽然也能完成一些简单的图形，或者使用 JavaScript 脚本实现，但是使用 canvas 可以在画布上自由的绘图。

20.3.2　使用 CSS 修改页面

CSS 允许用户为 HTML 5 文档中的各个部分指定格式，特别对静态的 HTML，CSS 的功能十分强大。此游戏的 CSS 样式表的名称为 main.css 文件。代码如下：

```
* {
    margin: 0;
    padding: 0;
```

```
    }
    html {
        background-attachment: fixed;
        background: #bda9a2;
        background: -moz-linear-gradient(top,   #bda9a2 0%, #90817a 47%, #6f6059 100%);
        background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#bda9a2), color-stop(47%,
#90817a), color-stop(100%,#6f6059));
        background: -webkit-linear-gradient(top,   #bda9a2 0%,#90817a 47%,#6f6059 100%);
        background: -o-linear-gradient(top,   #bda9a2 0%,#90817a 47%,#6f6059 100%);
        background: -ms-linear-gradient(top,   #bda9a2 0%,#90817a 47%,#6f6059 100%);
        background: linear-gradient(to bottom,   #bda9a2 0%,#90817a 47%,#6f6059 100%);
        filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#bda9a2', endColorstr='#6f6059',
GradientType=0 );
        height: 100%;
    }
    header {
        background-color:rgba(33, 33, 33, 0.9);
        color:#fff;
        display:block;
        font: 14px/1.3 Arial,sans-serif;
        height:50px;
        position:relative;
    }
    header h2{
        font-size: 22px;
        margin: 0px auto;
        padding: 10px 0;
        width: 80%;
        text-align: center;
    }
    header a, a:visited {
        text-decoration:none;
        color:#fcfcfc;
    }
    #scene {
        display: block;
        height: 700px;
        margin: 10px auto;
        position: relative;
        width: 700px;
    }
```

经过 CSS 样式表修改后，效果如图 20-6 所示。



图 20-6　CSS 修饰过的界面

20.3.3　JavaScript 编程

JavaScript 是一种脚本编程语音，提供了不少内置的函数来访问 HTML 文档的各个部分，包括 CSS 元素的样式。

另外，本游戏使用的是 Jquery 框架，Jquery 是继 prototype 之后又一个优秀的 JavaScript 框架。它是轻量级的 JS 库，兼容 CSS3，还兼容各种浏览器。jQuery 使用户能更方便地处理 HTML documents、events，实现动画效果。

其中，JavaScript 文件的内容和作用如下：

```
// 定义图像对象
var backgroundImage;
var oRocketImage;
var oExplosionImage;
var introImage;
var oEnemyImage;
var iBgShiftY = 9300; //水平宽度为 10000   - canvas 高度为 700
var bPause = true; // 游戏暂停
var plane = null; // 定义飞机对象
var rockets = []; // 定义飞机移动数组
var enemies = []; // 定义敌机数组
var explosions = []; // 定义飞机爆炸数组
var planeW = 200; // 定义飞机的宽度
var planeH = 110; // 定义飞机的高度
var iSprPos = 2; // 飞机的初始框架
```

```
var iMoveDir = 0; // 定义移动方向
var iEnemyW = 128; // 敌机宽度
var iEnemyH = 128; // 敌机高度
var iRocketSpeed = 10; // 初始飞机移动速度
var iEnemySpeed = 5; // 初始敌机的移动速度
var pressedKeys = []; // 定义按键数组
var iScore = 0; // 初始化分数
var iLife = 100; // 初始化生命值
var iDamage = 10; // 敌机撞击玩家的飞机后的损坏值
var enTimer = null; // 随机出现一个敌机
// 定义飞机函数对象
function Plane(x, y, w, h, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.image = image;
    this.bDrag = false;
}
// 定义攻击函数对象
function Rocket(x, y, w, h, speed, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.speed = speed;
    this.image = image;
}
// 定义敌机函数对象
function Enemy(x, y, w, h, speed, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.speed = speed;
    this.image = image;
}
// 定义爆炸函数对象
function Explosion(x, y, w, h, sprite, image) {
    this.x = x;
    this.y = y;
    this.w = w;
```

```
        this.h = h;
        this.sprite = sprite;
        this.image = image;
    }
    // 从 X 和 Y 之中随机地获取字母的函数
    function getRand(x, y) {
        return Math.floor(Math.random()*y)+x;
    }
    // 初始显示函数
    function displayIntro() {
        ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);
        ctx.drawImage(introImage, 0, 0,700, 700);
    }
    // 绘制主要的场景函数
    function drawScene() {
        if (! bPause) {
            iBgShiftY -= 2; // 飞机移动的范围
            if (iBgShiftY< 5) {
                bPause = true;
                // 绘制场景
                ctx.font = '40px Verdana';
                ctx.fillStyle = '#fff';
                ctx.fillText('Finish, your score: ' + iScore * 10 + ' points', 50, 200);
                return;
            }
            // 通过按键控制飞机
            processPressedKeys();
            // 清除绘图
            ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);
            // 绘制游戏背景
            ctx.drawImage(backgroundImage, 0, 0 + iBgShiftY, 700, 700, 0, 0, 700, 700);
            // 绘制游戏飞机
            ctx.drawImage(plane.image, iSprPos*plane.w, 0, plane.w, plane.h, plane.x - plane.w/2, plane.y -
plane.h/2, plane.w, plane.h);
            // 绘制爆炸效果
            if (rockets.length >0) {
                for (var key in rockets) {
                    if (rockets[key] != undefined) {
                        ctx.drawImage(rockets[key].image, rockets[key].x, rockets[key].y);
                        rockets[key].y -= rockets[key].speed;
                        // 如果攻击在屏幕范围之外，则忽略攻击
                        if (rockets[key].y< 0) {
```

```
                        delete rockets[key];
                    }
                }
            }
        }
        // 绘制爆炸
        if (explosions.length >0) {
            for (var key in explosions) {
                if (explosions[key] != undefined) {
                    // 显示爆炸效果
                    ctx.drawImage(explosions[key].image,  explosions[key].sprite*explosions[key].w,  0,
explosions[key].w, explosions[key].h, explosions[key].x - explosions[key].w/2, explosions[key].y - explosions[key].
h/2, explosions[key].w, explosions[key].h);
                    explosions[key].sprite++;
                    // 如果敌机被击毙，则消除爆炸对象
                    if (explosions[key].sprite >10) {
                        delete explosions[key];
                    }
                }
            }
        }
        // 绘制敌机
        if (enemies.length >0) {
            for (var ekey in enemies) {
                if (enemies[ekey] != undefined) {
                    ctx.drawImage(enemies[ekey].image, enemies[ekey].x, enemies[ekey].y);
                    enemies[ekey].y -= enemies[ekey].speed;
                    // 假如敌机在屏幕之外，则忽略敌机对象
                    if (enemies[ekey].y >canvas.height) {
                        delete enemies[ekey];
                    }
                }
            }
        }
        if (enemies.length >0) {
            for (var ekey in enemies) {
                if (enemies[ekey] != undefined) {
                    // 碰撞的爆炸效果
                    if (rockets.length >0) {
                        for (var key in rockets) {
                            if (rockets[key] != undefined) {
                                if  (rockets[key].y<  enemies[ekey].y  +  enemies[ekey].h/2  &&
```

```
rockets[key].x >enemies[ekey].x && rockets[key].x + rockets[key].w< enemies[ekey].x + enemies[ekey].w) {
                                        explosions.push(new          Explosion(enemies[ekey].x          +
enemies[ekey].w / 2, enemies[ekey].y + enemies[ekey].h / 2, 120, 120, 0, oExplosionImage));
                                        // 消除敌机和爆炸，然后增加 1 分
                                        delete enemies[ekey];
                                        delete rockets[key];
                                        iScore++;
                                    }
                                }
                            }
                        }
                        // 飞机的碰撞
                        if (enemies[ekey] != undefined) {
                            if (plane.y - plane.h/2< enemies[ekey].y + enemies[ekey].h/2 && plane.x -
plane.w/2< enemies[ekey].x + enemies[ekey].w && plane.x + plane.w/2 >enemies[ekey].x) {
                                explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w / 2,
enemies[ekey].y + enemies[ekey].h / 2, 120, 120, 0, oExplosionImage));
                               // 消除敌机和制作损坏
                                delete enemies[ekey];
                                iLife -= iDamage;
                                if (iLife<= 0) { // 游戏结束
                                    bPause = true;
                                    // 计算总得分
                                    ctx.font = '38px Verdana';
                                    ctx.fillStyle = '#fff';
                                    ctx.fillText('Game over, your score: ' + iScore * 10 + ' points', 25, 200);
                                    return;
                                }
                            }
                        }
                    }
                }
            }
            // 显示生命值和得分
            ctx.font = '14px Verdana';
            ctx.fillStyle = '#fff';
            ctx.fillText('Life: ' + iLife + ' / 100', 50, 660);
            ctx.fillText('Score: ' + iScore * 10, 50, 680);
        }
    }
    // 游戏过程中的按键函数
    function processPressedKeys() {
```

```
        if (pressedKeys[37] != undefined) { // 左键控制效果
            if (iSprPos >0) {
                iSprPos--;
                iMoveDir = -7;
            }
            if (plane.x - plane.w / 2 >10) {
                plane.x += iMoveDir;
            }
        }
        else if (pressedKeys[39] != undefined) { // 右键控制效果
            if (iSprPos< 4) {
                iSprPos++;
                iMoveDir = 7;
            }
            if (plane.x + plane.w / 2< canvas.width - 10) {
                plane.x += iMoveDir;
            }
        }
    }
    // 增加随机的敌机函数
    function addEnemy() {
        clearInterval(enTimer);
        var randX = getRand(0, canvas.height - iEnemyH);
        enemies.push(new Enemy(randX, 0, iEnemyW, iEnemyH, - iEnemySpeed, oEnemyImage));
        var interval = getRand(1000, 4000);
        enTimer = setInterval(addEnemy, interval); // 函数循环执行
    }
    // 主要的默认函数
    $(function(){
        canvas = document.getElementById('scene');
        ctx = canvas.getContext('2d');
        // 载入背景图片
        backgroundImage = new Image();
        backgroundImage.src = 'images/levelmap.jpg';
        backgroundImage.onload = function() {
        }
        backgroundImage.onerror = function() {
            console.log('Error loading the background image.');
        }
        introImage = new Image();
        introImage.src = 'images/intro.jpg';
        // 初始化敌机攻击效果
```

```
        oRocketImage = new Image();
        oRocketImage.src = 'images/rocket.png';
        oRocketImage.onload = function() { }
        // 初始化爆炸图片
        oExplosionImage = new Image();
        oExplosionImage.src = 'images/explosion.png';
        oExplosionImage.onload = function() { }
        // 初始化敌机
        oEnemyImage = new Image();
        oEnemyImage.src = 'images/enemy.png';
        oEnemyImage.onload = function() { }
        // 初始化玩家战机
        var oPlaneImage = new Image();
        oPlaneImage.src = 'images/plane.png';
        oPlaneImage.onload = function() {
            plane = new Plane(canvas.width / 2, canvas.height - 100, planeW, planeH, oPlaneImage);
        }
        $(window).keydown(function (evt){ // 手动控制飞机的事件
            var pk = pressedKeys[evt.keyCode];
            if (! pk) {
                pressedKeys[evt.keyCode] = 1; // 将按键添加进数组
            }
            if (bPause && evt.keyCode == 13) { // 按下 Enter 键效果
                bPause = false;
                // 开始游戏
                setInterval(drawScene, 30); // 循环绘制游戏场景
                // 添加一个敌机
                addEnemy();
            }
        });
        $(window).keyup(function (evt) {
    // 松开按键后的事件
            var pk = pressedKeys[evt.keyCode];
            if (pk) {
                delete pressedKeys[evt.keyCode];
    // 从数组中移除按键
            }
            if (evt.keyCode == 65) { // 按 A 键开始攻击
                rockets.push(new Rocket(plane.x - 16, plane.y - plane.h, 32, 32, iRocketSpeed, oRocketImage));
            }
            if (evt.keyCode == 37 || evt.keyCode == 39) {
                // revert plane sprite to default position
```

```
                if (iSprPos >2) {
                    for (var i = iSprPos; i >= 2; i--) {
                        iSprPos = i;
                        iMoveDir = 0;
                    }
                } else {
                    for (var i = iSprPos; i<= 2; i++) {
                        iSprPos = i;
                        iMoveDir = 0;
                    }
                }
            }
        });
        // 当游戏开场条件准备好后开始游戏
        introImage.onload = function() {
            displayIntro();
        }
    });
```